ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Douglas Neves de Jesus

DESENVOLVIMENTO DE UM APLICATIVO DE RECOMENDAÇÃO DE ARTIGOS CIENTÍFICOS PARA MATERIAIS DIDÁTICOS

Palmas

2012

Douglas Neves de Jesus

DESENVOLVIMENTO DE UM APLICATIVO DE RECOMENDAÇÃO DE ARTIGOS CIENTÍFICOS PARA MATERIAIS DIDÁTICOS

Trabalho apresentado como requisito parcial da disciplina Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pela Professora Mestre Parcilene Fernandes de Brito.

Palmas

2012

Douglas Neves de Jesus

DESENVOLVIMENTO DE UM APLICATIVO DE RECOMENDAÇÃO DE ARTIGOS CIENTÍFICOS PARA MATERIAIS DIDÁTICOS

Trabalho apresentado como requisito parcial da disciplina Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pela Professora Mestre Parcilene Fernandes de Brito.

Aprovada em 22 de junho de 2012.

BANCA EXAMINADORA

_____

Prof. M.Sc. Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Fabiano Fagundes

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Jackson Gomes de Souza

Centro Universitário Luterano de Palmas

Palmas

2012

À minha irmã.

Agradecimentos

Agradeço à minha mãe, Valdenira, por sempre ter me dado todo apoio necessário, abdicando muitas vezes do seu tempo por mim. Por ter me ensinado a sempre respeitar o próximo e tratar a todos, sem exceção, com educação; preceitos que foram fundamentais para a formação do meu caráter. Por sempre ter me dado a oportunidade de pensar por conta própria em minhas escolhas, mas sempre me aconselhando quando necessário. Pela amizade, amor e companheirismo que sempre poderei contar.

Agradeço ao meu pai, José Petrônio, por sempre ter feito o máximo que pode para não deixar faltar nada em nossa família. Por me ensinar que o esforço é sempre recompensado e que podemos ter êxito em qualquer profissão que escolhida, dês de que goste do que faça. Por ter me proporcionado todo o apoio que precisei para concluir meus estudos.

Agradeço à minha irmã, Samantha, por ter me dado a honra de termos nascidos no mesmo dia, muitas vezes dizendo que fui "o presente de seu aniversário". Por ter me apoiado diversas vezes em nossa família, me mostrando que eu poderia sempre contar com o que precisasse; as diversas vezes em que te visitava só para dar um "olá" sempre ajudaram quando faltavam forças para seguir em frente, me mostrando que eu não estava sozinho em uma cidade longe dos pais. Por sempre ter corrido atrás de seus sonhos e fazendo de tudo para que dêem certo, me ensinando que devemos sempre ser persistentes e jamais desistir. Por sempre ter feito de tudo para unir nossa família, com seu jeito alegre e otimista de viver a vida, nos mostrando que independente da situação não podemos desistir do que nos faz feliz. Pelo carinho que sempre tive, embora muitas vezes não merecesse.

Agradeço ao meu sobrinho, Alexandre, pelas inúmeras vezes que me motivou para continuar os estudos. Pelas ligações no meio do serviço só para falar comigo. Por ser mais um motivo de que eu nunca devo desistir. O tio te ama muito.

Agradeço a todos meus amigos da faculdade por terem me apoiado nos momentos difíceis e por termos compartilhado experiências, até então, novas para mim.

Agradeço a Deisinha, pela amizade, pelos diversos momentos compartilhados na faculdade e por me incentivar a concluir o curso. As diversas risadas nos corredores da faculdade, saídas a noite, musicas cantadas em inglês (e que inglês ein? rs) e vários outros momentos no decorrer do curso me fizeram gostar ainda mais de ti. Agradeço pela paciência e pelo nosso jeito peculiar de fingir que nada aconteceu após uma briga.

Luane, tivemos vários momentos bons e ruins nesse período que nos conhecemos, mas isso só me mostra o quanto você importante pra mim, tanto como companheira de faculdade quanto

amiga. Agradeço pelas horas de descontração, sustos nos corredores da faculdade, trabalhos em casa (sim, aquela também foi a melhor batata que eu já comi kkkkk).

Naara... minha companheira em diversos trabalhos da faculdade e até profissionais. Como esquecer as noites sem dormir fazendo proict? Das constantes ligações te acordando pra ir a ULBRA estudar? Dos momentos de alegria e também de frustração com os sistemas do estágio? Agradeço a faculdade por ter me permitido te conhecer, sua paz de espirito (rá, não podia deixar de citar isso Naara rsrs) me faz te admirar cada dia mais, continue sempre assim, essa garota que "faz o bem sem olhar quem".

Agradeço ao Roneylson pela amizade e companheirismo nesses quatro anos de convivência. Foi com você que compartilhei vários momentos bons tanto no curso quanto na vida. Crescemos e amadurecemos juntos. Se lembra da nossa primeira conversa (eu, você e Cleydiane) "séria" por papel na aula do Fabiano? Des daquele dia eu percebi que teríamos uma grande amizade, e assim está sendo. Temos tantas historias que protagonizamos que daria pra escrever um livro (e que livro kkk). Sei que posso contar contigo pro que precisar, e que apesar das diferentes opiniões em diversos momentos, aqui estamos nós, firmes e fortes (be strong).

Agradeço ao Rodrigo pelas diversas noites de conversa, pela amizade e por me mostrar que grandes amizades podem surgir quando a gente menos espera. Foi você que me ajudou diversas vezes quando eu estava sem animo para terminar o TCC, e mesmo sem saber, seu jeito divertido de protagonizar a vida me alegravam madrugada a fora. Lembra quando combinamos de ir pra academia e você disse que no caminho começou a tocar "Too Little, Too Late"??? eu fico rindo sozinho disso até hoje kkkkk.

Agradeço a Bianca pela paciência em minha enorme ausência nessa reta final. Sua amizade é muito importante pra mim, e vou levar ela para a vida toda. Obrigado por sempre acreditar em minha capacidade, mesmo eu não merecendo, isso me estimulava cada vez mais a concluir o curso.

Agradeço aos demais amigos que conquistei no decorrer do curso, me perdoem se na correria esqueci alguém: Charles, Cil, Cleydiane, Cris, Douglas Brito, Lucas, Mayanne, Rafael (leafar) e Ricardo.

Agradeço aos amigos que conquistei no trabalho: Beth, Carol, Fagner, Marcio, Philipe, Telma e William, em especial para a Telma, Beth e Fagner, por todo apoio que recebi durante o desenvolvimento do TCC, muitas vezes me cedendo tempo quando necessitava.

Agradeço aos Professores.

Andrez, por sua forma sutil de acabar com a felicidade momentânea dos alunos (vulgo: dar uma tirada). Acho que até hoje a turma toda se lembra do "Andrez: você quer escolher as questões da prova? Deise: quero. Andrez: Mas não pode" kkkkkkk. Agradeço por ter conhecido esse excelente profissional que é.

Cristina, como esquecer a "melhor professora de instrumentalização"?? Infelizmente não pude ter aula dessa matéria contigo, mas tive o prazer de ter aprendido diversas outras. Você é um exemplo de disciplina e seriedade, seu jeito imparcial de dar aulas é, pra mim, uma de suas maiores qualidades. Agradeço a amizade que pude contar em diversos momentos, ao ombro amigo quando precisei.

Edeilson, só tivemos uma única aula no decorrer do curso, mas foi o suficiente pra conhecer a grande pessoa que é. Te considero um grande amigo e, apesar da mania de querer me bater quando me ve rsrs, sei que posso contar com seus ensinamentos.

Fabiano, Thilfa, Amigo. Você foi mais que um professor, foi um grande amigo que conquistei; Acreditou em mim quando eu não merecia, me incentivou sempre a conquistar meus objetivos, me mostrou que a vida não é fácil. Segurou meu choro quando eu mais precisei e, acredite, eu nunca vou me esquecer disso. Quando eu pensei em trancar o curso, você me disse "Falta pouco, não desista" e graças a isso estou hoje aqui, formado. Tenho você como um exemplo de vida e é em você que me inspiro quando começo a achar que as coisas não darão certo para mim. Obrigado por tudo.

Fernando. Meu orientador de estágio e eterno orientador. Graças a você aprendi a melhorar minha escrita, e foi graças aos seus ensinamentos que pude escrever o TCC de forma menos trabalhosa. Agradeço a amizade e os diversos momentos divertidos compartilhados dentro e fora da sala de aula.

Jack, o pequeno grande homem =). Você me deu várias boas oportunidades de aprendizado no decorrer do curso e, embora eu não soubesse as aproveitar, foi graças a elas que pude superar os desafios da profissão. Agradeço pelas palavras nas diversas vezes que precisei... nos e-mails, conversas de madrugada, banquinho do lado do prédio... Você é, sem duvidas, um grande amigo pra mim, e agradeço por também me permitir ser seu amigo.

Madia, a "professora querida"; De fato você realmente é uma professora querida, basta perguntar a qualquer aluno que tenha tido ou não uma aula com você. Sempre preocupada com os alunos e muita muitas vezes sendo rígida quando necessário. Agradeço a amizade e por, juntamente com a Cleydiane, Sara e Edeilson, sermos uma família na volta da Ulbra para casa.

Parcilene... Quando entrei no curso fiquei intrigado com o que outros alunos diziam: “A Parcilene é sem coração”; no começo fiquei meio apreensivo, até porque a primeira aula do semestre era com ela (Algoritmos e Programação I), mas logo fui vendo a pessoa que você realmente é. Quem te conhece, mesmo que pouco, sabe que você não é nada do que os boatos dizem, muito pelo contrário, você tem um coração enorme, um coração de mãe. Minha mãe no curso. Eu como filho (desnaturado filho rs) te deixei por várias vezes frustrada, mas mesmo assim você sempre me acolhia. Me ensinou que não devemos complicar as coisas e que as dificuldades na verdade não passam de barreiras que a nossa própria mente cria; que pensar sem agir não tem efeito algum. Me lembro de cada e-mail seu... uns enormes, outros mais ainda... mas em todos eles, apesar das “broncas”, você sempre foi esperançosa. No decorrer do curso eu pensei várias vezes que não conseguiria concluir, que você fosse desistir de mim; mas que mãe desiste do filho? Você acreditou que eu conseguiria, e apesar dos meus “surtos”, se manteve disposta até o final, me ajudando sempre. Eu digo, com certeza, que não teria concluído sem sua ajuda; seus ensinamentos foram fundamentais para a minha formação e serei eternamente grato a ti por isso. Além da Parcilene mãe também tenho a Parcilene amiga, que soube me ouvir quando precisei e, embora algumas coisas não tenham a agradado, me aconselhou na medida em que achava certo. Agradeço a preocupação nos momentos difíceis, as conversas divertidas tanto pelo talk quanto pessoalmente e ao café e comida diversas vezes “roubados” em sua sala. Obrigado por tudo. =)

Agradeço também as demais pessoas que me ajudaram a concluir o curso, sem vocês essa jornada seria mais difícil e tediosa.

SUMÁRIO

RESUMO

Uma das maiores problemáticas em se trabalhar com informações textuais é a dificuldade em identificar quais informações são, de fato, úteis para um determinado propósito. A partir disto surgiram técnicas de Recuperação da Informação e de Recomendação para auxiliar a busca de informações relevantes. Tendo como domínio a rede social acadêmica Konnen, que está sendo desenvolvida no CEULP/ULBRA, o presente trabalho tem como objetivo a criação de um aplicativo de recomendação de artigos científicos para o módulo de materiais didáticos da rede social acadêmica. Esse aplicativo irá utilizar as informações do material didático (título, descrição e *tags*) e, a partir disso, recuperar artigos científicos de um repositório virtual previamente definido. Após a obtenção dos artigos, inicia-se o processo de recomendação, utilizando os conceitos da filtragem baseada em conteúdo para recomendar os artigos que possuem um maior nível de similaridade com o material didático. Por fim, serão apresentadas as etapas do desenvolvimento do aplicativo de recomendação.

PALAVRAS-CHAVE: Recuperação da Informação, *WebCrawler* e Sistema de Recomendação

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

FBC	Filtragem Baseada em Conteúdo
IDF	*Inverse Document Frequency*
RI	Recuperação da Informação
SGBD	Sistema de Gerenciamento de Banco de Dados
SR	Sistemas de Recomendação
TF	*Term Frequency*

1 INTRODUÇÃO

Com o aumento da facilidade do acesso à internet, houve uma crescente diversidade e quantidade de conteúdo disponibilizado no meio virtual, através de blogs, *wikis* e sites diversos. Com esse crescente aumento de informação, a busca manual por conteúdo passou a se tornar trabalhosa e exaustiva; até mesmo com a utilização de motores de busca ainda existe certa dificuldade em localizar algo relevante (Aires, 2005).

A Recuperação da Informação (RI) foi criada então para minimizar esse esforço do usuário em buscar por uma informação que seja relevante para seu interesse. Já que, à medida que o usuário requisita um resultado a partir de uma consulta, sistemas de RI aplicam etapas para que este resultado seja o mais próximo do desejável, fazendo com que o usuário obtenha informações que atendam suas necessidades ou até mesmo uma informação não esperada, mas que também seja útil ao usuário. Entretanto, mesmo com essa estratégia de recuperação, ainda existe certa dificuldade em relacionar essas informações a outras e sugerir conteúdo ao usuário, necessitando de algum processo que avalie o quão similar são essas informações, já que as preferências do usuário também podem ser utilizadas no momento da recuperação. Com isso surge a necessidade de utilizar outros processos que auxiliem nessa sugestão de conteúdo ao usuário, dentre eles estão os Sistemas de Recomendação (SR).

Sistemas de Recomendação são responsáveis por diminuir o tempo na busca por produtos ou conteúdo, pois atuam diretamente em recomendar aquilo que se espera pelo usuário, utilizando de preferências explicitadas diretamente pelo usuário ou informações parametrizadas para realizar a recomendação (BARCELLOS et al, 2007, p. 3). SR também atuam de forma a relacionar um conteúdo a outro, formando assim uma rede de recomendação, em que um conteúdo está relacionado a outros conteúdos similares, gerando uma ligação entre eles.

Atualmente, no CEULP/ULBRA (Centro Universitário Luterano de Palmas), está em desenvolvimento a rede social acadêmica Konnen. A Konnen irá disponibilizar diversos conteúdos (materiais didáticos) para seu público alvo, no entanto esses conteúdos não possuem relação, no que tange a recomendação, com outros materiais externos à rede, como artigos científicos, necessitando assim de uma forma com que esse material (artigos científicos) seja inserido no contexto do aluno como um conteúdo a mais para estudo.

A partir dessa problemática, este trabalho tem como objetivo oferecer um mecanismo que utilize técnicas de Recuperação da Informação para buscar artigos científicos que possuam certo grau de similaridade com o conteúdo (materiais didáticos) da rede social acadêmica Konnen. Utilizando as informações do conteúdo de material didático da Konnen (título, descrição e *tags*), serão aplicadas técnicas de Recomendação para verificar o grau de similaridade entre os artigos científicos e os materiais didáticos e os recomendar para o usuário de acordo com sua similaridade.

Nas próximas seções serão apresentados os conceitos necessários para o desenvolvimento do trabalho. Inicialmente serão definidos conceitos de Recuperação da Informação (2.1), assim como as etapas necessárias para a execução deste processo. Logo após será introduzido o conceito de Sistemas de Recomendação (2.2) e sua importância para a utilização no trabalho. Serão apresentados também os materiais e métodos (3) utilizados, assim como as etapas do aplicativo de recomendação em resultados e discussões (4). Por fim as considerações finais (5), abordando também possíveis melhorias e sugestões como trabalhos futuros.

2 REFERENCIAL TEÓRICO

Esta seção apresentará os conceitos necessários para a compreensão do funcionamento do módulo de recomendação de artigos científicos para a rede social acadêmica Konnen. Os conceitos envolvidos serão Recuperação da Informação, *WebCrawler* e Sistemas de Recomendação.

2.1. Recuperação da Informação

De acordo com Gonzalez & Lima (2001, p. 2), "a essência da Recuperação da Informação (RI) consiste na busca de documentos relevantes a uma dada consulta que expressa a necessidade de informação do usuário". Dessa forma, entende-se que RI não se limita apenas as condições de busca explicitadas pelo usuário, como uma instrução realizada em um Sistema de Gerenciamento de banco de dados (SGBD), mas sim na sua real necessidade, recuperando possíveis informações relevantes.

Essas consultas, se relacionadas as preferências do usuário, podem agregar um maior valor – no que tange a relevância – ao usuário, visto que ao relacionar uma maior quantidade de informações, as possibilidades de se recuperar o que deseja são maiores, ainda que tal fato não exclua também a recuperação daquilo que não é de interesse do usuário. As próximas seções irão detalhar o processo que torna essa recuperação da informação mais precisa.

Segundo Ferneda (2003, p. 15), os sistemas de RI "devem representar o conteúdo dos documentos do corpus[1] e apresentá-los ao usuário de uma maneira [...] que satisfazem total ou parcialmente à sua necessidade de informação". Os documentos são toda e qualquer fonte de informação que agregue valor a um determinado assunto. Com o advento do computador, os documentos não se limitaram mais a apenas uma representação textual, mas também a outros tipos de formatos que também tenham um grau de relevância para o usuário, como vídeos, imagens, sons e qualquer outro tipo de representação de informação.

Essa representação de conteúdo consiste em organizar os documentos através da técnica de indexação (que será detalhada mais a frente), tornando o acesso à informação mais eficiente. Essa etapa visa classificar os documentos por meio de características para que os mesmos tenham uma "identidade"; isso permite que, ao se buscar um documento que tenha

[1] Conjunto de textos

uma determinada característica, outros documentos com a mesma característica sejam associados àquele primeiro, retornando assim possíveis documentos similares.

O usuário atua diretamente junto ao sistema de RI através de uma expressão de busca, como por exemplo, buscar por um determinado termo em campo de busca de um *website*. Essa expressão de busca visa retornar ao usuário tanto as informações que ele deseja quanto as que ele nem ao menos imaginou que seriam úteis já que sistemas de RI trazem outros documentos relacionados. O usuário também pode especificar o que é útil para o mesmo a partir do resultado daquela expressão de busca, iniciando um processo de "pontuação" de informação, em que é explicitada a relevância daquele documento para que, em uma consulta futura, ao se recuperar um documento com aquela expressão, o que não for útil seja descartado (Figura 1).

Figura 1 - pontuador de relevância

A Figura 1 exemplifica esse processo de pontuação, no qual o usuário informa uma expressão de busca e são retornadas consultas relacionadas à expressão. Caso o usuário avalie que um resultado possui maior importância que outro, ele pode especificar que aquela informação é útil, como na imagem em que é possível alterar a ordem dos resultados clicando no ícone da seta para cima.

Para relacionar a representação com a expressão de busca tem-se a "função de busca", que é responsável por comparar esse conjunto de características do conteúdo a fim de retornar ao usuário os documentos que mais se assemelham a sua necessidade. Ferneda (2003, p. 18) complementa ainda que apesar de um documento estar presente na representação, não quer dizer que ele seja útil ao usuário. Para isso existem três fatores que devem ser considerados: diversos termos de busca; o termo da busca pode não ser de um contexto que atenda a necessidade do usuário; o documento pode já ter sido recuperado pelo usuário ou ser antigo demais para consulta.

Aires (2005, p. 9) especifica o processo de RI de um sistema em quatro etapas:

> i) representar cada documento em uma forma que possa ser "compreendida" pelo computador, ii) interpretar as consultas fornecidas, iii) comparar as consultas interpretadas com o conjunto de documentos indexados, e iv) apresentar os resultados de forma adequada à necessidade do usuário.

O usuário irá fornecer uma consulta que será interpretada pelo sistema de RI, assim é realizada uma comparação entre a consulta interpretada e os documentos indexados, apresentando ao usuário o resultado. O usuário ainda interage com o sistema informando a relevância dos documentos recuperados, de forma a tornar o processo de recuperação mais eficiente (como observado na Figura 1).

Sendo assim, mesclando os conceitos e definições vistos anteriormente, o processo de recuperação da informação pode se dar na forma da seguinte estrutura (Figura 2):

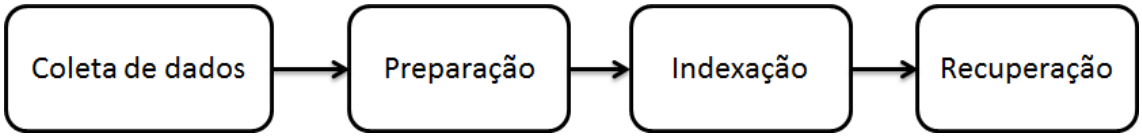


Figura 2 - Processo de Recuperação da Informação

A próxima seção irá detalhar cada etapa do processo de RI visto na figura anterior (Figura 2).

2.1.1. Etapas

Essa seção irá detalhar cada etapa presente na Figura 1, demonstrando a importância de cada uma para o processo de Recuperação da Informação.

2.1.1.1. Coleta de dados

Essa etapa é responsável por coletar os documentos que serão utilizados no processo de Recuperação da Informação. A coleta pode ser feita de forma manual ou automática (BRANSKI, 2004):

- **Manual**: explicitados diretamente de um local, onde já se tem os documentos armazenados. Utilizada em sistemas gerais de RI, como em sistemas de bibliotecas.
- **Automática**: Utilizada em sistemas Web, em que é realizada uma coleta do conteúdo das páginas utilizando *crawlers* (robôs).

2.1.1.1.1. WebCrawler

Um *crawler* pode ser compreendido como um método ou *script* responsável por fazer buscas em páginas web e indexar seu conteúdo para uma pós-utilização (AIRES, 2005, p. 166). Cada *crawler* é projetado especificamente para o domínio que será utilizado, tendo suas etapas e rotinas definidas a partir da necessidade da aplicação. Como por exemplo, caso uma aplicação necessite de informações diárias de uma determinada página web, o sistema de *crawler* é projetado para realizar a rotina de coleta diariamente.

Sua importância se estende desde o processo de indexação das páginas (através de URLs), o qual é utilizado por motores de busca[2] para facilitar o processo de busca por conteúdo na web, até a aquisição do conteúdo contido na página, que será o utilizado no desenvolvimento deste trabalho.

Segundo Coelho e Azevedo (2008, p. 3) existem três problemas encontrados por mecanismos de *crawler*: "o seu grande volume [de conteúdo], a rápida modificação de conteúdos e a geração dinâmica de páginas". Alguns desses problemas, como a modificação do conteúdo, podem ser resolvidos com configurações de rotina do *crawler*, que consiste em mapear o conteúdo em um intervalo de tempo menor, fazendo com que o conteúdo não fique tão desatualizado.

Outro problema na utilização de *crawlers* é o número de requisições que ele faz ao servidor, podendo ocupar boa parte da banda do proprietário do site. A partir disso foi criado um protocolo chamado *robots.txt*, que consiste em especificar comandos a fim de limitar as ações do *crawler*, podendo bloquear o acesso total ou parcial ao conteúdo do site, permitindo, assim, que um conteúdo privado não seja acessado (THELWALL e STUART, 2006).

A arquitetura básica de um *crawler* pode ser compreendida da seguinte forma (Figura 3):

[2] Sistemas projetados para localizar informações

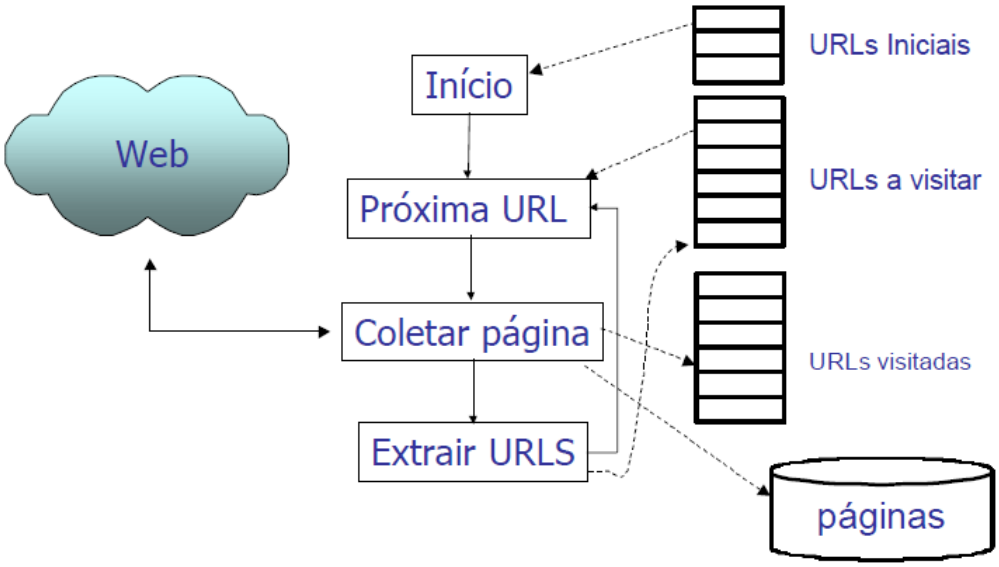


Figura 3 - Arquitetura do *crawler* (SILVA e MOURA, 2002, p. 4)

Como observado na Figura 3, em primeiro lugar, o *crawler* irá receber uma lista de URLs iniciais que serão utilizadas para a coleta; sequencialmente irá selecionar cada URL da lista e iniciar o processo de coleta, fazendo uma requisição ao servidor que hospeda a página, para coletar seu conteúdo; feito isso, a URL em questão será adicionada a uma lista de URLs visitadas. O conteúdo adquirido dessa página é persistido e as URLs encontradas são adicionadas em uma lista de URLs a visitar; o *crawler* volta para a etapa em que seleciona uma URL até que a lista de URLs inicias tenha chagado ao fim; em seguida inicia a coleta do conteúdo das páginas da lista de URLs a visitar. Nessa etapa, os dados não possuem estrutura ou organização, logo é necessário ainda passar para uma etapa de preparação, para que seja excluídas as informações irrelevantes, otimizando o processo de recuperação. A próxima seção irá detalhar essa etapa.

2.1.1.2. Preparação

Nessa etapa todo conteúdo indesejado é retirado do documento. Isso é necessário para que a filtragem da informação ocorra de forma mais precisa, visto que uma parte do conteúdo de uma mensagem pode ser dispensável no momento da busca. Para isso existem algumas técnicas herdadas da mineração de texto (*text mining*) que preparam o conteúdo para a posterior indexação. Monteiro, Gomes e Oliveira (2006, p.79) definem algumas técnicas, como: Correção Ortográfica, *Stemming* e *StopWords*; Corrêa (2011, p.7) complementa ainda que também pode ser utilizada a frequência de palavras no texto na etapa de preparação. A seguir é apresentado um maior detalhamento das técnicas da etapa de preparação.

- **Correção Ortográfica**: valida as palavras do texto de acordo com um vocabulário pré-definido. Esse processo também evita vícios de escrita frequentemente encontrados com o advento da internet, como "vc" e "pq" que seriam corrigidos para "você" e "porque".
- ***Stemming***: essa etapa consiste em eliminar todas as variações de escrita das palavras (gerúndio, plural, prefixos e sufixos), reduzindo a palavra ao seu radical. Dessa forma, cada palavra estará associada ao seu elemento "raiz", reduzindo assim o número de termos no documento (FERNEDA, 2003, p. 84). A Tabela 1 exemplifica esse processo.

Tabela 1 - *Stemming* em palavras

Termo	Stemming
maioria	maior
maiores	maior

Como visto na tabela anterior (Tabela 1), os sufixos "ia" e "es" foram retirados da palavra, formando assim o radical "maior" que está associado àqueles dois termos.

- ***StopWords***: de acordo com Corrêa (2011, p.7) , essa etapa é importante pois é responsável por eliminar do texto palavras irrelevantes. O processo consiste em criar uma lista de palavras (*StopList*) que devem ser retiradas do texto, como preposições, artigos, conjunções, adjetivos e advérbios (BARION e LAGO, 2008, p.28). A Tabela 2 exemplifica esse processo.

Tabela 2 - StopWords em documentos

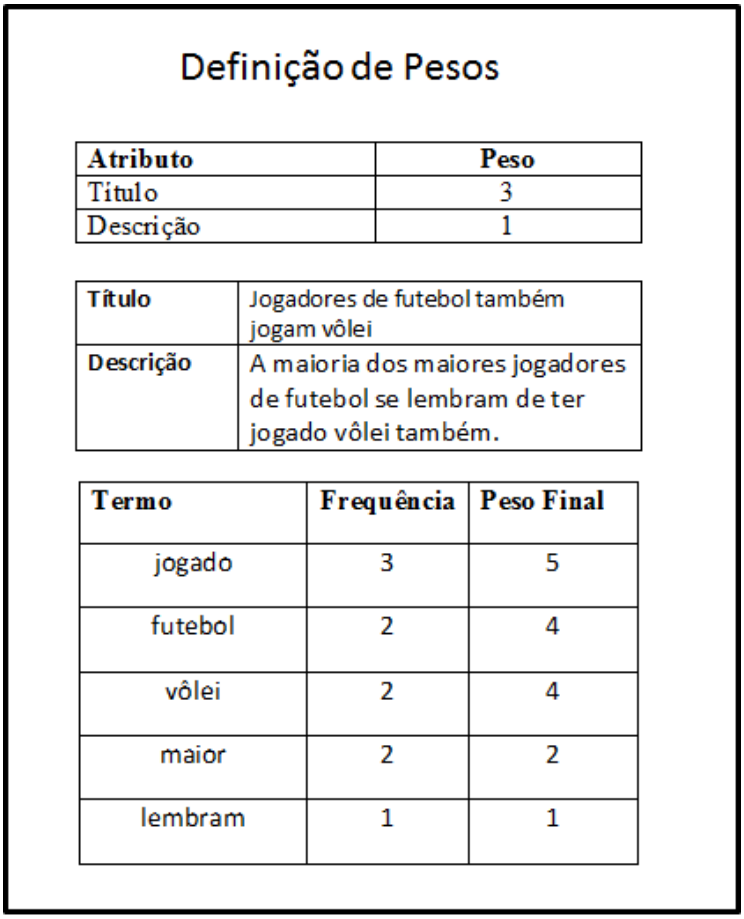
Documento Original
A maioria dos maiores jogadores de futebol se lembram de ter jogado vôlei também
Documento com *StopWords*
Maioria maiores jogadores futebol lembram jogado vôlei
StopList
a
dos
de
se
ter
também

- **Frequência de Palavras**: consiste em contar o número de aparições de uma mesma palavra no documento analisado (CORRÊA, 2011, p.7). A Tabela 3 exemplifica esse processo.

Tabela 3 - Frequência de palavras

Documento	Palavras	Frequência
A maioria dos maiores jogadores de futebol se lembram de ter jogado vôlei também	maior	2
	jogado	2
	futebol	1
	lembram	1
	vôlei	1

A Tabela 3 exemplifica o processo de frequência de palavras. No documento é aplicado o processo de *stemming* e *stopwords* e com isso são contadas as palavras iguais e é feita uma lista contando a ocorrência das palavras repetidas no texto. Com essa lista de frequência são aplicados “pesos” a cada palavra, fazendo com que certas palavras possuam uma relevância maior que outras dependendo da sua importância. Logo, é necessário definir quais atributos devem ser levados em consideração ao definir o peso (Figura 4).



Definição de Pesos

Atributo	Peso
Título	3
Descrição	1

Título	Jogadores de futebol também jogam vôlei
Descrição	A maioria dos maiores jogadores de futebol se lembram de ter jogado vôlei também.

Termo	Frequência	Peso Final
jogado	3	5
futebol	2	4
vôlei	2	4
maior	2	2
lembram	1	1

Figura 4 - Definição dos Pesos

Como exemplificado na figura anterior (Figura 4), foi criado um documento fictício contendo os atributos "título" e "descrição", no qual foram atribuídos pesos para eles, fazendo com que o atributo "título" possua uma maior importância em relação ao atributo "descrição", pois título possui termos mais específicos da característica do documento. Logo em seguida é separado o conteúdo de cada atributo e montada uma lista de termos com base em sua frequência. Por fim são multiplicados os devidos pesos a cada ocorrência do termo nos conteúdos, no qual os termos do atributo "título" terão suas ocorrências multiplicadas por três e os termos do atributo "descrição" terão suas ocorrências multiplicadas por um, ao final são somados os pesos dos respectivos termos, formando assim um peso final para cada termo.

2.1.1.3. Indexação

Para Vieira e Corrêa (2010, p. 03) a etapa de indexação é responsável pela "construção da representação do conteúdo dos documentos através da atribuição de termos (palavras-chave) ou códigos de indexação que serão úteis posteriormente na recuperação desses documentos". Esses termos são responsáveis por criar uma estrutura de recuperação de informação no

documento, tornando as informações antes avulsas em informações recuperáveis, pois são criadas palavras-chaves referenciando informações. Essas palavras-chave (índices) irão referenciar documentos que possuam contextos semelhantes, funcionando, também, como um filtro a fim de minimizar o tempo de sua busca (Figura 5).

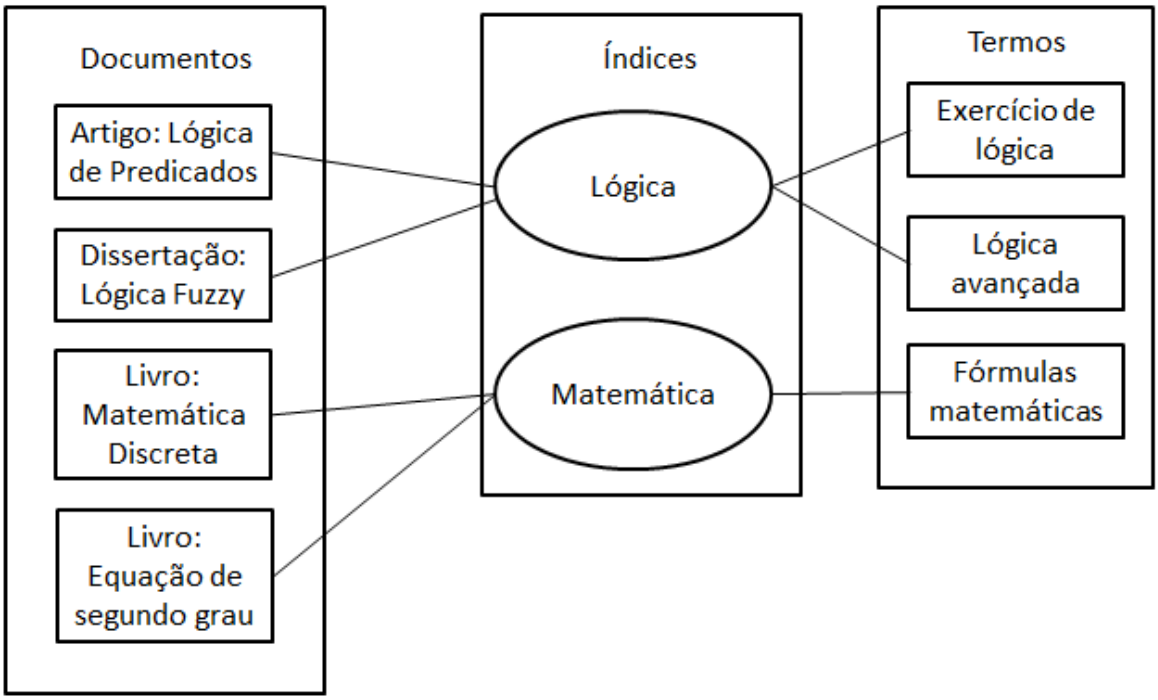


Figura 5 - Indexação de Documentos

Como observado na figura anterior (Figura 5), os índices são criados de forma a relacionar termos que façam parte de um mesmo contexto, tornando o índice uma palavra-chave que representa as expressões de buscas. Após formar os índices, são relacionados os termos e os documentos ao respectivo índice deste contexto. Essa etapa é importante no processo de recuperação, pois visa acessar de forma mais rápida documentos que possam ser semelhantes ao que o usuário necessita, desconsiderando assim documentos que não possuem relevância, pois documentos distintos estarão indexados em índices distintos.

2.1.1.4. Recuperação

A etapa de recuperação consiste na utilização de técnicas para obter um resultado o mais próximo do desejado pelo usuário. Para Ferneda (2003, p.18) “a eficiência de um sistema de recuperação de informação está diretamente ligada ao modelo que o mesmo utiliza”. Um modelo é a representação de estratégias de busca de documentos relevantes (CARDOSO, 2004, p. 2). Dessa forma, deve-se escolher um modelo que atenda a necessidade do domínio

proposto, no qual agregue um resultado mais satisfatório. A próxima seção irá explanar alguns modelos

2.1.1.4.1. Modelos

Os modelos são responsáveis por determinar estratégias de recuperação de documentos a partir de uma consulta; cada modelo considera que um documento *i* possui um conjunto de termos *ij* e que esses termos possuem um peso $W_{i,j} \geq 0$ (CARDOSO, 2004, p. 2). Os modelos clássicos mais utilizados são: modelo booleano e o modelo espaço vetorial:

- **Modelo Booleano**

No modelo booleano as consultas são criadas a partir de expressões booleanas, utilizando os operadores lógicos AND, OR e NOT. Nesse modelo os termos de busca são considerados relevantes ou não relevantes, não existindo assim um meio termo, como em outros modelos. As figuras a seguir (Figura 6, Figura 7, Figura 8 e Figura 9) exemplificam melhor a utilização desses operadores lógicos.

Figura 6 - Representação do resultado de uma expressão booleana conjuntiva (AND) (FERNEDA, 2003, p.22)

Como observado na Figura 6, o operador AND é responsável por retornar os documentos que foram indexados tanto pelo termo t_1 quanto pelo termo t_2, representando a área cinza na figura, ou seja, o campo relativo à intersecção dos conjuntos.

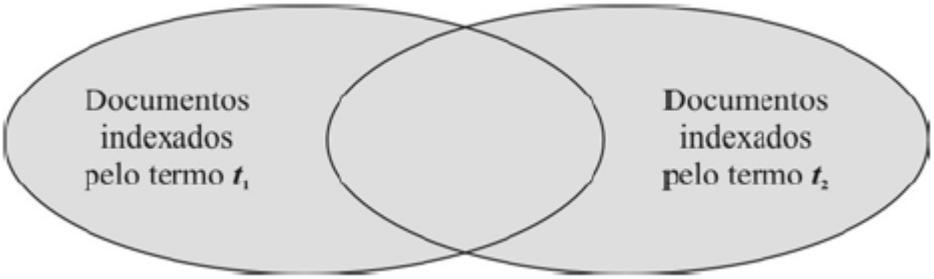


Figura 7 - Resultado de uma busca booleana disjuntiva (OR) (FERNEDA, 2003, p.22)

O operador OR é responsável por retornar todos os documentos que foram indexados pelo termo t_1 e todos os documentos que foram indexados pelo termo t_2, retornando assim todos os documentos referentes à união dos dois conjuntos, como observado na área cinza (Figura 7).

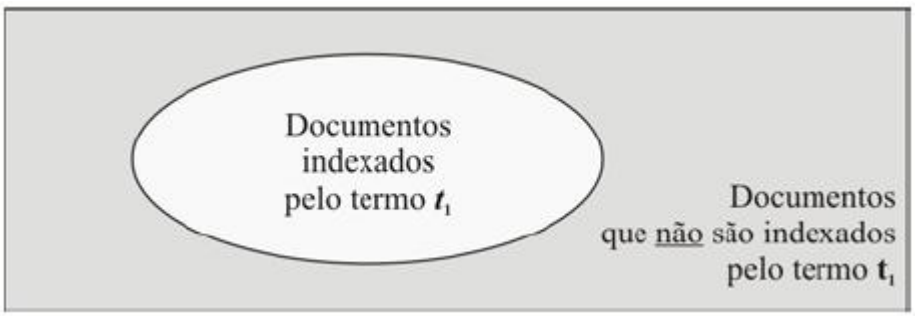


Figura 8 - Resultado de uma busca negativa (NOT) (FERNEDA, 2003, p.23)

Como observado na figura anterior (Figura 8), o operador NOT é responsável por retornar todos os documentos que não foram indexados pelo termo t1, representando a área cinza na figura.

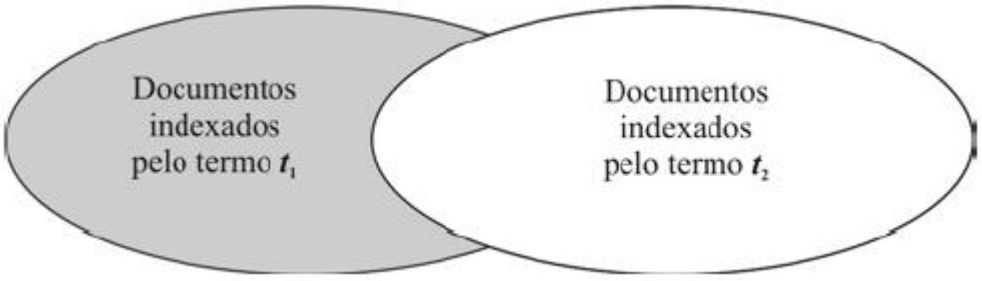


Figura 9 - Resultado de uma busca booleana com o operador NOT (FERNEDA, 2003, p.23)

Uma expressão que contém o operador NOT entre dois termos resultará em todos os documentos que foram indexados pelo termo t_1, mas que não foram indexados pelo termo t_2, representado a área cinza na figura (Figura 9).

- **Modelo Espaço Vetorial**

O modelo espaço vetorial consiste em representar os documentos textuais em vetores, sendo cada termo do documento um elemento do vetor. Para cada termo presente em uma *query*[3] e em um documento são atribuídos pesos que são utilizados para representar o vetor em um espaço Euclidiano (Figura 10) (CARDOSO, 2004, p. 3).

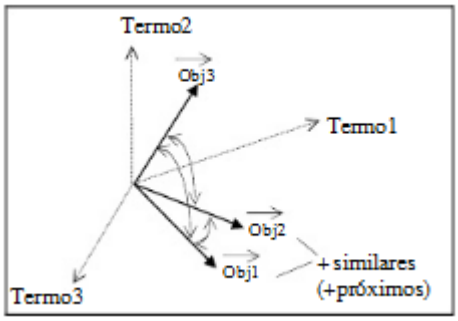


Figura 10 - O modelo espaço-vetorial (WIVES, 2002, p. 39)

O ângulo que é formado entre esses vetores é chamado de θ. Ao calcular o cosseno desse ângulo θ é possível definir o nível de similaridade entre os vetores, resultando na seguinte equação:

$$\cos(\theta) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c \times \vec{w}_s}{\|\vec{w}_c\| \times \|\vec{w}_s\|} = \frac{\sum_{i=1}^{k} w_{i,c}\, w_{i,s}}{\sqrt{\sum_{i=1}^{k} w_{i,c}^2}\sqrt{\sum_{i=1}^{k} w_{i,s}^2}}$$

O resultado do cálculo varia entre 0 e 1, sendo 0 uma total dissimilaridade e 1 uma total similaridade entre os documentos. A sessão 2.2.1 irá apresentar uma melhor explicação e utilização do modelo espaço vetorial, assim como a demonstração da fórmula do cálculo de pesos dos termos.

As técnicas de recuperação são responsáveis por montar estratégias de recuperação de documentos, no entanto ao exibir esses documentos ao usuário levando em consideração suas preferências é necessária a utilização de outra abordagem em conjunto, que é a de Sistemas de Recomendação. A próxima seção irá detalhar mais sobre o funcionamento dos Sistemas de Recomendação.

[3] Pode ser entendido como “Consulta” em português.

2.2. Sistema de Recomendação

"Os Sistemas de Recomendação (SR) são utilizados para [...] recomendar itens que podem ser produtos, serviços e/ou conteúdos, de acordo com suas necessidades e interesses." (BARCELLOS et al, 2007, p. 3). Dessa forma, a essência de um Sistema de Recomendação consiste em recomendar ao usuário aquilo que reflete sua real necessidade, seja ela perceptível ou não para ele. Ao se recomendar um item, é necessário definir quais parâmetros serão levados em consideração, seja, por exemplo, ao consumir produtos em um comercio eletrônico ou por número de acessos a uma determinada página de uma categoria, havendo ou não interação do usuário e suas preferências. A Figura 11 exemplifica uma recomendação.

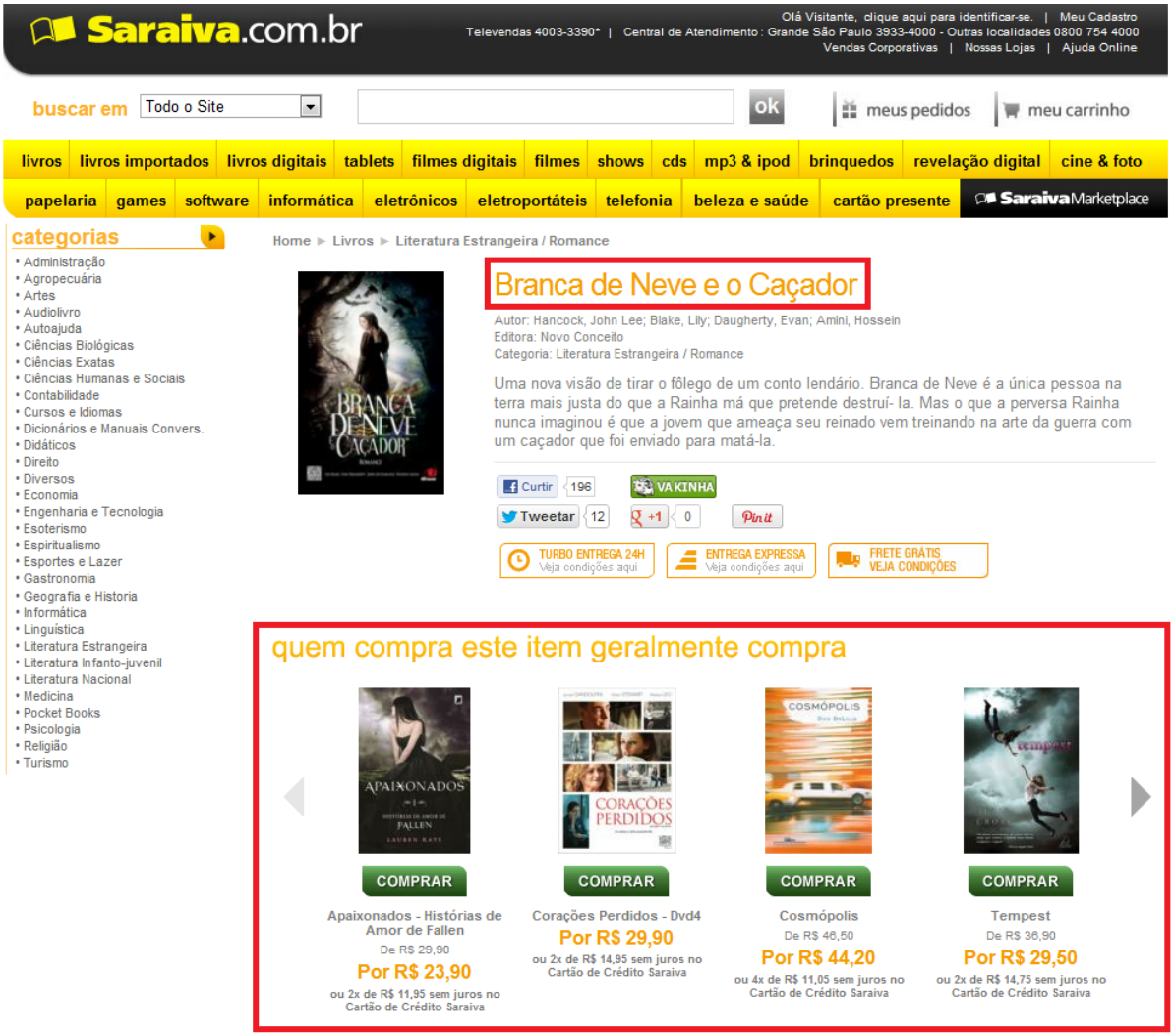


Figura 11 - Recomendação de produtos

Como observado na Figura 11, ao ser consultado o livro "Branca de Neve e o Caçador" na página da livraria Saraiva, o site recomenda alguns outros produtos que possam ser do interesse do usuário. Isso é possível, pois o site faz uma busca dos produtos que foram

comprados pelo usuário após a compra do livro “Branca de Neve e o Caçador” e, a partir disso, recomenda os que são semelhantes ao livro, nesse caso outros livros também de literatura.

Existem diversas técnicas de recomendação com a finalidade de atender as necessidades do usuário e dentre elas está a Filtragem baseada em Conteúdo (FBC). Como o presente trabalho utilizará as informações de materiais didáticos (título, *tags* e descrição) ao invés das preferências do usuário, a FBC é a técnica mais adequada a esse contexto, visto que a recomendação será feita no momento em que o material é cadastrado, não tendo assim influência de outros usuários; esse processo será mais bem detalhado na seção 4.

2.2.1. Filtragem Baseada em Conteúdo (FBC)

“A filtragem baseada em conteúdo emprega a comparação entre o conteúdo dos itens [...]. Essa abordagem tem suas origens nas técnicas empregadas em sistemas de recuperação da informação” (TORRES, 2003 apud PEREIRA, 2007, p. 13). Dessa forma, são utilizadas características dos itens, como: *tags*, descrição, título, para gerar a recomendação. Esses itens são retirados de um conjunto de documentos e considera apenas a *query* utilizada na consulta, e não diretamente as preferências do usuário (BAEZA-YATES e RIBEIRO-NETO, 1999, p. 21). Essas preferencias podem ser obtidas, como por exemplo, ao comprar um produto em um comercio eletrônico, no qual o usuário não precisou informar que possui interesse por produtos semelhantes, mas por ter comprado é possível que produtos que possuem características parecidas também o interesse.

Sendo assim, a Filtragem Baseada em Conteúdo não necessita que o usuário interaja diretamente com o sistema para realizar as recomendações, pois à medida que o conteúdo de um item na base de dados for semelhante ao conteúdo de novos itens que irão surgir, esses possuem uma maior probabilidade de serem recomendados para o usuário que consumiu o primeiro item. As Figuras 12 e 13 exemplificam o funcionamento da FBC.

Figura 12 - Acesso ao site da NETSHOES como visitante

Como observado na Figura 12, um usuário novo (visitante) faz seu primeiro acesso ao site com o intuito de adquirir alguns itens; como é seu primeiro acesso e não foi consumido nenhum item, o site não consegue filtrar as preferências do usuário, não podendo assim recomendar produtos que sejam do seu interesse (pois, embora o usuário não tenha especificado diretamente seu interesse, ao adquirir um produto é possível mensurar seu interesse por produtos semelhantes), exibindo quaisquer tipos de produtos na página inicial.

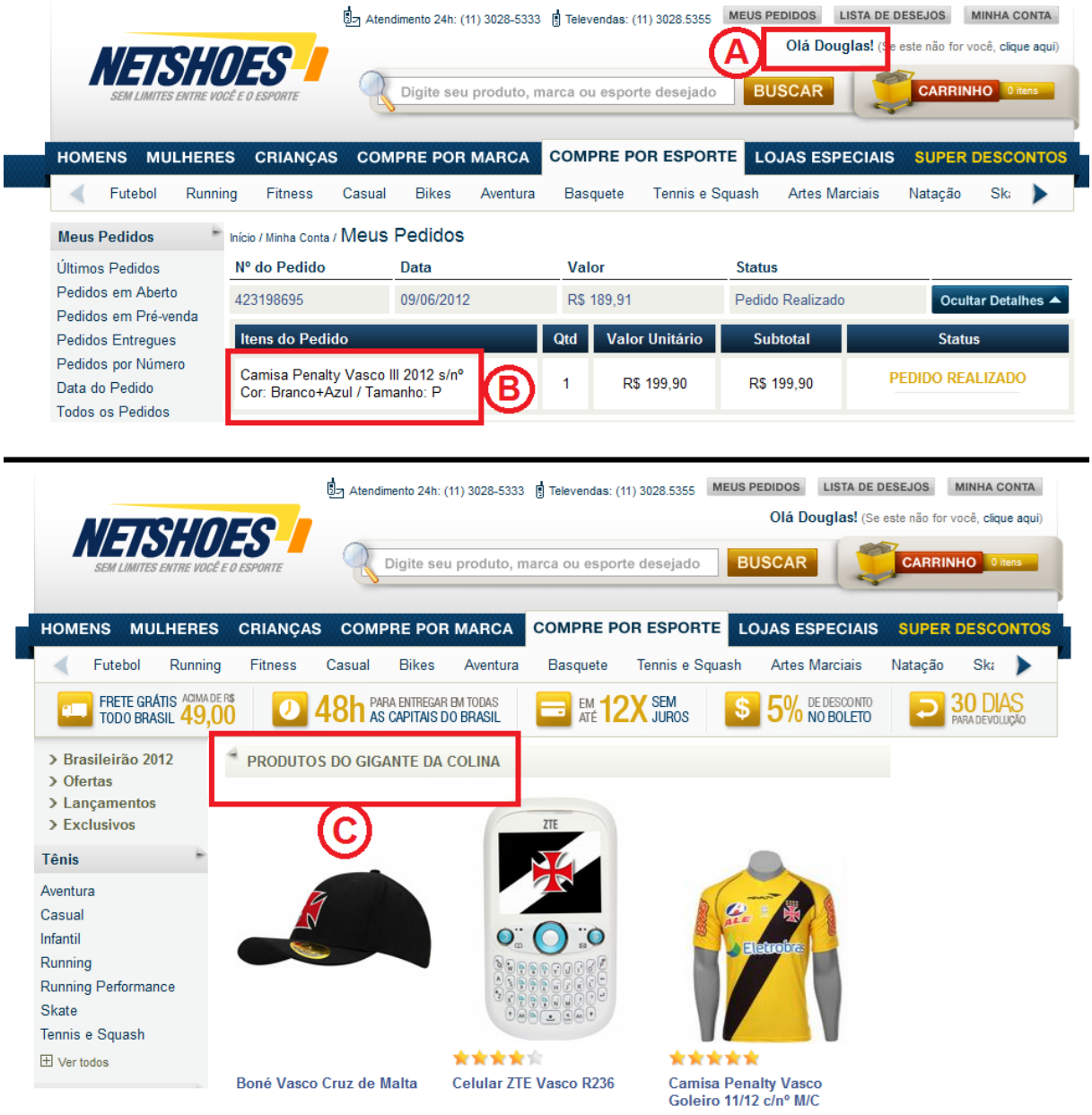


Figura 13 - Acesso ao site da NETSHOES com um usuário

Após se identificar no site com um usuário registrado - Douglas (Figura 13 - A), o sistema procura os produtos que foram comprados pelo usuário – neste caso, produtos de um determinado time de futebol (Figura 13 – B). Após recolher as informações pertinentes aos produtos consumidos pelo usuário, o sistema utiliza a técnica de FBC para recomendar produtos que sejam similares aos que o usuário já tenha adquirido, exibindo assim na página inicial recomendação de produtos do time preferido do usuário (Figura 13 - C).

A FBC costuma ser utilizada em domínios onde os itens possuem informações em forma textual (PEREIRA, 2007, p.13), assim, todos os itens que possuem informações

semelhantes aos que foram consumidos pelo usuário poderão ser recuperados e recomendados.

Para implementar essa técnica, primeiramente é necessária a utilização do algoritmo TF-IDF, que consiste em verificar a relevância de um conjunto de documentos em relação a um termo específico de um documento. Esses dois coeficientes são representados da seguinte forma: TF – frequência do termo no documento a ser comparado; IDF - frequência inversa do termo no conjunto de documentos recuperados (PEREIRA, 2007, p.13).

Para calcular a frequência do termo no documento é utilizada a seguinte equação:

$$tf(t,d) = \frac{frequencia}{MaximaFrequencia}$$

Nessa equação, o peso *tf*(*t*, *d*) de um termo *t* no documento *d* pode ser calculado dividindo a frequência com que o termo é encontrado no documento pela frequência do termo que mais aparece no documento, obtendo assim o peso daquele determinado termo. Exemplificando, para calcular o peso de um termo X, que possui três ocorrências em um documento Y, que possui outro termo Z com seis ocorrências, o cálculo da frequência de termos é apresentado da seguinte forma:

$$tf(X,Y) = \frac{3}{6} = 0{,}5$$

O IDF, *Inverse Document Frequency*, representa a pouca incidência de um determinado termo no documento, resultando em um termo com pouca relevância. Para calcular o IDF é utilizado o seguinte cálculo matemático:

$$idf_t = \log \frac{TotalDeDocumentosNoCorpus}{NumeroDeDocumentosQueContemOTermo(t)}$$

Nessa fórmula, a frequência inversa é calculada aplicando o *log* da divisão entre o total de documentos contidos no *corpus* pelo número de documentos do *corpus* em que um determinado termo ocorre. Exemplificando, para calcular a frequência inversa de um termo X, que possui ocorrência em seis documentos, em um total de sessenta documentos, é feito o cálculo da seguinte forma:

$$idf = \log \frac{60}{6} = 1$$

Após realizar o cálculo da frequência do termo (*tf*) e a frequência inversa do termo (*idf*) é possível calcular o TF-IDF, resultando na multiplicação desses dois coeficientes:

$$\text{TF-IDF} = \text{tf} \times \text{idf}$$

Com o peso de cada termo é possível aplicar o cálculo do cosseno para medir a similaridade entre os documentos. O cálculo a seguir representa como é feita essa similaridade:

$$similaridade(q,d) = \frac{\sum_{i=1}^{k} W_{i,q} W_{i,d}}{\sqrt{\sum_{i=1}^{k} W_{i,q}^2} \sqrt{\sum_{i=1}^{k} W_{i,d}^2}}$$

O cosseno do ângulo entre os documentos pode ser calculado dividindo a soma do produto dos pesos dos termos nos documentos *q* e *d* pela multiplicação entre as raízes dos somatórios dos quadrados de cada peso em $W_{i,q}$ e $W_{i,d}$. A Figura 13 apresenta um exemplo com dois documentos e seus respectivos termos.

Documento q			
Termo	Termo 1	Termo 2	Termo 3
Peso	0,4	0,5	1

Documento d			
Termo	Termo 1	Termo 2	Termo 3
Peso	0,4	1	0,5

Figura 14 - Pesos de cada termo dos documentos

Após calcular o peso de cada termo dos documentos *q* e *d* (Figura 13) utilizando o calculo do TF-IDF é possível utilizar o cálculo do cosseno para avaliar o nível de similaridade que o documento *q* possui com o documento *d*. O cálculo a seguir demonstra como é obtida essa avaliação:

$$\cos(q,d) = \frac{(0{,}4 \times 0{,}4) + (0{,}5 \times 1) + (1 \times 0{,}5)}{\sqrt{0{,}4^2 + 0{,}5^2 + 1^2} \times \sqrt{0{,}4^2 + 1^2 + 0{,}5^2}} = 0{,}82$$

Como observado na fórmula anterior, o resultado do nível de similaridade entre o documento *q* e o documento *d* foi avaliado em 0,82 em um intervalo entre 0 (nenhuma

similaridade) e 1 (maior similaridade), tornando-se assim documentos potencialmente semelhantes.

Um dos problemas da FBC é a necessidade de que o usuário tenha adquirido algum item do domínio, como por exemplo, ao comprar uma musica em um *website*, pois não é possível calcular a similaridade se não houver algum item que seja de preferência do usuário para utilizar como parâmetro, visto que a FBC se baseia em comparar a similaridade entre documentos. Também não é possível gerar recomendações de contextos diferentes ao usuário, pois seria necessário que o usuário consumisse um item desse outro contexto para a FBC recomendar itens semelhantes a ele.

A seção seguinte irá apresentar os materiais e métodos utilizados para o desenvolvimento desse trabalho.

3 MATERIAIS E MÉTODOS

Nessa seção são apresentados os materiais utilizados e a metodologia adotada para o desenvolvimento do trabalho.

3.1. Local e Período

O desenvolvimento do presente trabalho ocorreu no complexo laboratorial do Centro universitário Luterano de Palmas – CEULP/ULBA e em residência própria, no período de 2012/1, como requisito das disciplinas "Trabalho de conclusão de Curso I" e "Trabalho de Conclusão de Curso II".

3.2. Materiais

Para o desenvolvimento deste trabalho foram utilizados recursos de hardware próprios e diversas fontes para montar o referencial teórico, como: dissertações de mestrado, teses de doutorado, revistas, artigos e outros materiais complementares.

Para a implementação do mecanismo de recomendação foi utilizada a linguagem de programação C#, juntamente com a IDE *Microsoft Visual Studio* 2010:

- Linguagem C#: o C# (C Sharp) é uma linguagem de programação orientada a objetos, fortemente tipada e de propriedade da *Microsoft*. Surgiu em 2001 e tem como um dos seus principais compiladores o .NET *Framework*. Demais informações podem ser adquiridas em: < http://msdn.microsoft.com/pt-br/vstudio/hh388566.aspx >
- *Microsoft Visual Studio* 2010: é uma ferramenta que auxilia o processo de desenvolvimento de software, criada em 2010 com o objetivo de ser uma das IDE's (*Integrated Development Environment* – Ambiente Integrado de Desenvolvimento) mais completas. Possui recursos que auxiliam na identificação de erros, depuração no código-fonte, testes, geração automática de código e outros. Demais informações podem ser adquiridas em: < http://www.microsoft.com/visualstudio/pt-br/ >

3.3. Metodologia

Inicialmente foi necessário entender o escopo do trabalho (considerando o contexto no qual ele está inserido) para uma definição melhor das etapas do aplicativo. Para isso, foram feitas reuniões presenciais e por *e-mail* com a orientadora a fim de prever algumas situações no desenvolvimento do trabalho.

Logo após, foi necessário também conhecer o domínio utilizado, para isto foi feita uma reunião com o coordenador do setor que desenvolve a rede social acadêmica para um melhor esclarecimento quanto a quesitos da implementação da rede e de seu funcionamento.

O aplicativo será desenvolvido para servir como uma ferramenta de recomendação da rede social acadêmica Konnen, sendo ligado ao módulo "conteúdo" da rede que é responsável pela inserção de qualquer tipo de conteúdo (imagens, vídeos, arquivos, etc.). As informações do conteúdo que serão utilizadas pelo aplicativo são: titulo, descrição e tags, dessa forma, para qualquer conteúdo que possua esses atributos, o aplicativo terá capacidade de gerar uma recomendação, visto que esse serão os parâmetros utilizados pra comparação entre os materiais didáticos e os artigos científicos.

Para desenvolver o aplicativo, foi necessário primeiramente construir o referencial teórico, de forma a obter um melhor conhecimento dos conceitos envolvidos no trabalho: Recuperação da Informação, *WebCrawler* e Sistema de Recomendação. Esse estudo possibilitou uma maior compreensão das etapas existentes na Recuperação da Informação e quais dessas etapas que seriam de fato utilizadas. Também foram feitos testes manuais do cálculo do cosseno, com o intuído de compreender de fato como é seu funcionamento para auxiliar no momento de codificação.

Foi necessário realizar uma busca por repositórios virtuais de publicações científicas para definir qual repositório seria o utilizado no trabalho. Os critérios de escolha se basearam tanto na facilidade de se extrair as informações do repositório, quanto no grau de confiabilidade dos responsáveis, assim como também em um maior número de publicações cientificas para se obter uma maior variedade de artigos.

No início do presente trabalho foi definido que o repositório utilizado seria o InfoCiência[4], um repositório virtual de artigos na área da computação, pois foi pensado em testar o funcionamento da recomendação em um domínio mais específico. No entanto no desenvolvimento do trabalho o repositório ficou indisponível por uma grande quantidade de tempo, necessitando de uma nova busca para encontrar outro repositório.

Dessa forma, optou-se por utilizar o SciELO[5], tanto por sua quantidade de artigos disponibilizados – aproximadamente 377 mil artigos – quanto por possuir publicações de diversas outras áreas, não somente computação.

[4] <http://infociencia.info> Acesso em: 25 Jun. 2012.
[5] < http://www.scielo.org> Acesso em: 27 Jun. 2012.

Por fim, foram definidas as etapas do aplicativo de recomendação, relacionando os conceitos vistos anteriormente na revisão de literatura com sua codificação. Essas etapas estão descritas passo a passo na próxima seção.

4 RESULTADOS E DISCUSSÃO

Nessa seção serão apresentadas as etapas necessárias para o desenvolvimento do aplicativo de recomendação de artigos científicos, assim como os resultados obtidos com a utilização do cálculo do cosseno. O desenvolvimento foi possível simulando um ambiente de cadastro de conteúdo, uma vez que a rede social acadêmica Konnen ainda está em fase de desenvolvimento e sua linguagem de programação é diferente da utilizada neste trabalho.

A seção a seguir irá demonstrar a arquitetura do Konnen, para que seja possível visualizar em que contexto o trabalho será inserido e de que forma ele irá atuar na rede social.

4.1. Arquitetura do Konnen

A rede social acadêmica Konnen, que está sendo desenvolvida para o CEULP/ULBRA, possui uma arquitetura de "múltiplas camadas e [...] permite que sejam criados aplicativos, com o objetivo de fornecer um modelo de extensões capaz de adicionar funcionalidade sob demanda para o sistema, usuários e grupos de usuários" (SOUZA, *et al*, 2012, p. 5). A Figura 15 apresenta a estrutura dos aplicativos do Konnen.

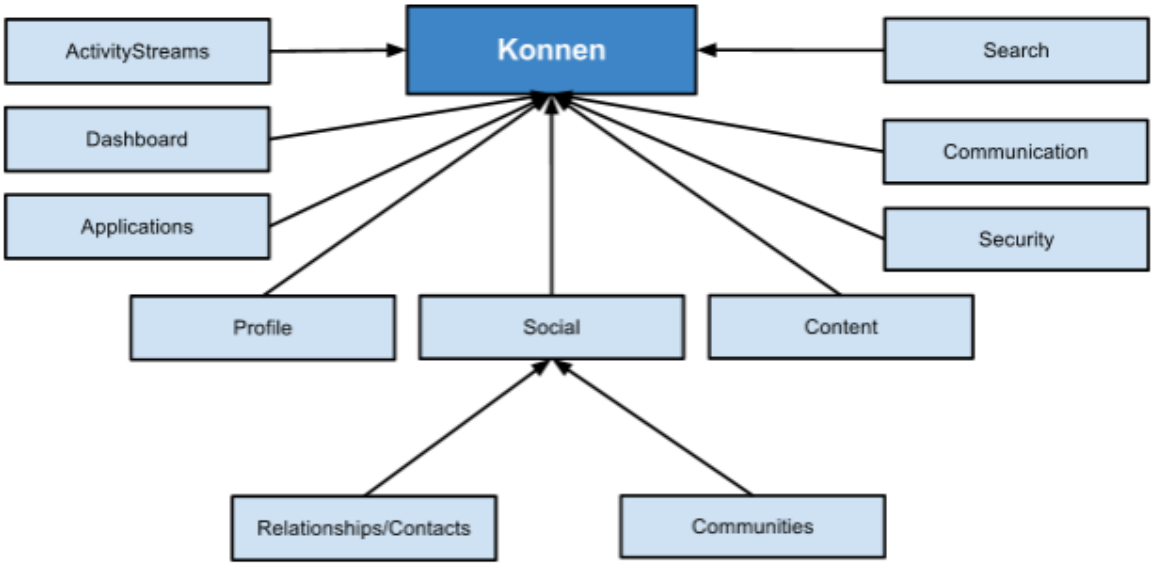


Figura 15 - Aplicativos (subsistemas) do Konnen (SOUZA, *et al*, 2012, p. 3)

Como observado na Figura 15, o Konnen possui uma estrutura de aplicativos que permite a adição de funcionalidades novas, agregando subsistemas à rede. O módulo que será

utilizado no trabalho será o "*content*", pois é o único que, de fato, será necessário para gerar as recomendações, visto que o conteúdo dos materiais didáticos (título, descrição e *tags*) será adquirido através desse módulo.

O projeto de pesquisa desenvolvido pelos responsáveis pelo Konnen "Aprendizagem Organizacional Através de uma Rede de Gestão de Conhecimento" demonstra essa interação entre a rede social e a recomendação de conteúdo na plataforma de aprendizagem organizacional (Figura 16).

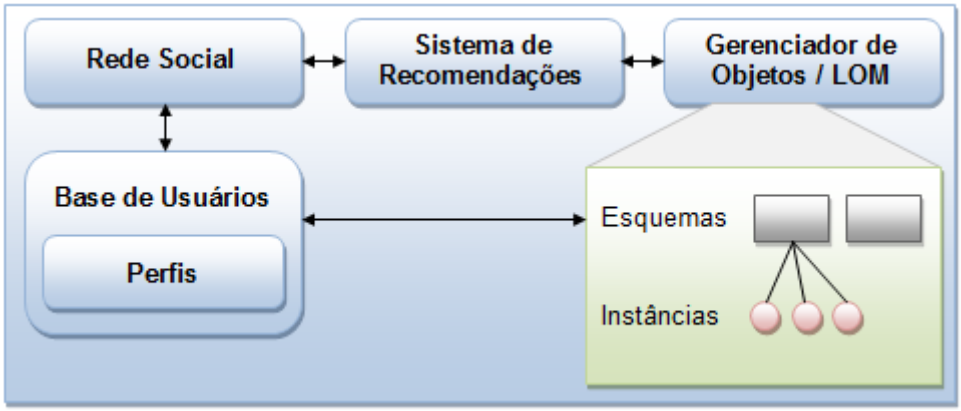


Figura 16 - Arquitetura da Plataforma (SOUZA, *et al*, 2012, p. 3)

Na figura 16 é possível observar que o Sistema de Recomendação é um módulo que interage com a rede social e com o Gerenciador de Objetos/LOM[6], recomendando objetos de acordo com o conhecimento produzido pela rede. O aplicativo desenvolvido neste trabalho representa uma parte desse módulo, ficando responsável por recomendar artigos científicos a materiais didáticos da rede. A seção seguinte apresentará as etapas e o funcionamento do mecanismo de recomendação de artigos científicos.

4.2. Aplicativo de Recomendação

O aplicativo de recomendação atua como um agente externo à rede, recebendo as informações do material didático (título, descrição e *tags*) e retornando uma lista de artigos potencialmente semelhantes ao material didático em questão. A figura a seguir (Figura 17) apresenta a arquitetura do aplicativo de recomendação.

[6] Módulo responsável pelo gerenciamento de esquemas (descrições de dados que podem ser criadas pelo usuário). (SOUZA, *et al*, 2012, p. 4)

Figura 17 - Arquitetura do Aplicativo de Recomendação

A estrutura do aplicativo de recomendação ilustrado na figura Figura 17 funciona da seguinte forma:

1. Inicialmente o Professor (ou outro usuário autenticado com permissão de cadastro de material) cadastra o material didático na rede social acadêmica Konnen com as devidas informações necessárias (título, descrição e *tags*).
2. A rede social se comunica com o aplicativo de recomendação passando as informações do material didático cadastrado.
3. O aplicativo utiliza as informações do material didático para aplicar conceitos vistos anteriormente de recuperação da informação, montando uma *string* de busca e iniciando o processo de *crawler*. É importante observar que o *crawler* é realizado somente no momento em que se cadastra um material didático, dessa forma se houver um acréscimo de artigos no repositório em uma data posterior a data do cadastro do material didático, esses artigos não farão parte da recomendação, pois não há uma atualização das recomendações.
4. O *WebCrawler* realiza uma requisição no servidor do repositório escolhido e coleta as informações dos artigos recuperados.
5. Os artigos são retornados para o aplicativo de recomendação através da requisição.
6. O aplicativo utiliza técnicas de Recomendação (Filtragem Baseada em Conteúdo) para ordenar a lista de artigos de acordo com sua similaridade com o material didático.
7. A rede social recebe a lista de artigos e os vincula ao material didático, que serão posteriormente visualizados na página de materiais didáticos.

O diagrama a seguir (Figura 18) apresenta a estrutura que será utilizada no desenvolvimento do aplicativo.

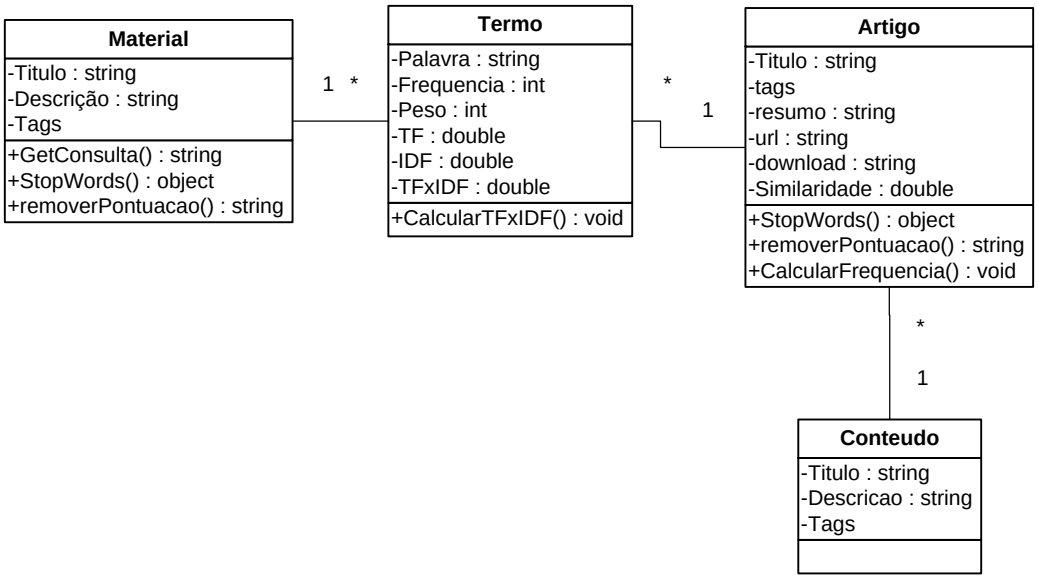


Figura 18 - Diagrama de Classes dos elementos utilizados no aplicativo

Como observado na Figura 18, foi necessário criar uma classe “Material” que será responsável por representar o material didático cadastrado no Konnen, e uma classe “Artigo” para representar os artigos científicos que serão obtidos do repositório. Cada uma dessas classes possui um relacionamento com a classe “Termo”, que é a lista de termos obtidas através dos atributos “titulo”, “descrição” e “tags”. A classe “Conteudo” representa o conteúdo da rede social, que será relacionado ao Artigo no final do processo de recomendação, possibilitando assim a ligação entre o conteúdo da rede e a recomendação.

A seção a seguir apresentará um melhor detalhamento de todas as etapas envolvidas no funcionamento do aplicativo.

4.2.1. Etapas

Esta seção irá apresentar cada etapa responsável pelo funcionamento do aplicativo, bem como sua relação com cada conceito visto no referencial teórico.

4.2.1.1. Simulação do Ambiente

A rede social Konnen possui um módulo de cadastro de conteúdo, ainda em desenvolvimento, no qual é possível extrair as informações necessárias para realizar a recomendação (Figura 19).

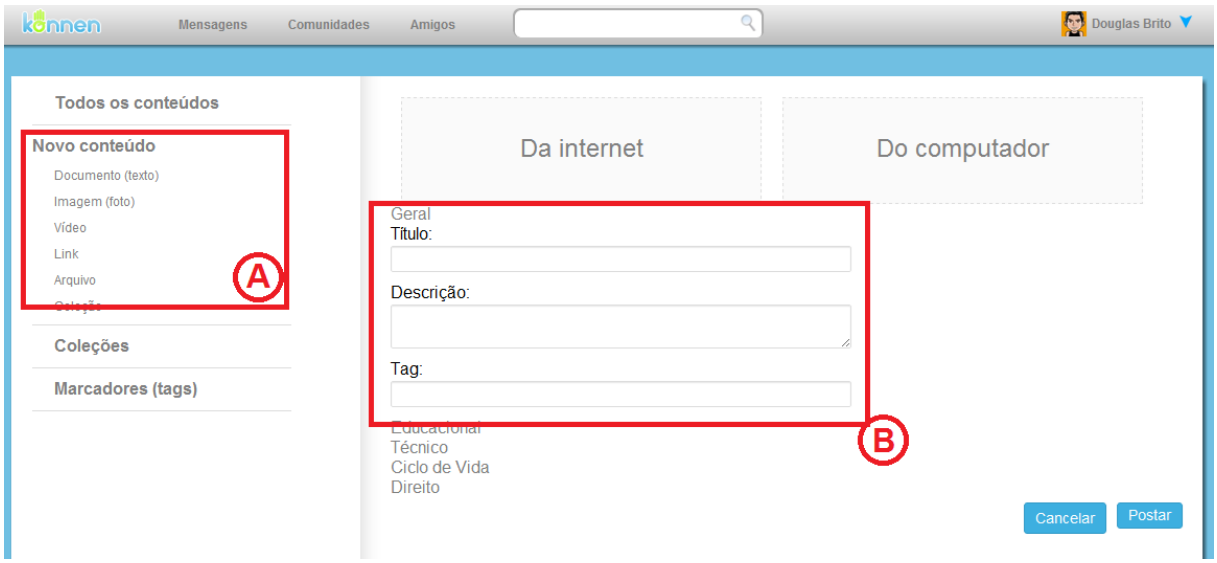


Figura 19 - Tela de cadastro de conteúdo

Como observado na figura anterior (Figura 19 – A), a página de cadastro de conteúdo possibilita que usuários compartilhem materiais de diferentes tipos (documento (texto), imagem (foto), vídeo, *link* e arquivo), passando informações como: título, descrição e *tags* (Figura 19 – B). O Konnen foi desenvolvido em uma linguagem de programação diferente da utilizada no desenvolvimento do aplicativo de recomendação, impossibilitando a utilização da tela de cadastro do Konnen para a chamada dos métodos do aplicativo devido a não integração das linguagens; para tanto, foi necessário então criar uma pagina genérica no ambiente de desenvolvimento do aplicativo contendo essas informações, para que seja possível extrair o conteúdo do material e executar os testes. A figura a seguir (Figura 20) apresenta a página criada para testar o aplicativo.

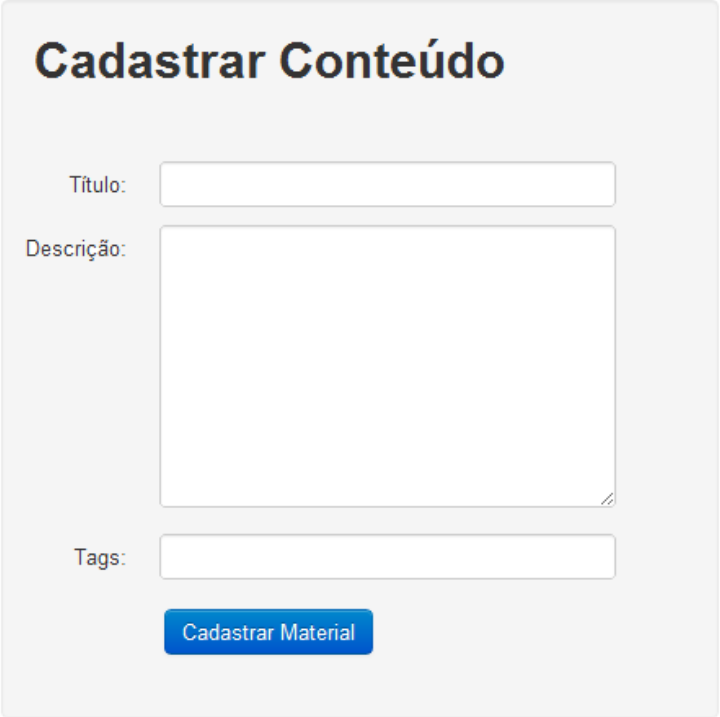


Figura 20 - Tela simulando o cadastro de conteúdo

A figura 20 demonstra a tela genérica de cadastro de conteúdo; sua utilização atende os requisitos para recomendar conteúdo, visto que não é necessário utilizar o ambiente do Konnen para realizar os testes, pois as informações de título, descrição e *tags* podem ser as mesmas em ambos os ambientes, tornando assim confiável o resultado da recomendação. Após preparar o ambiente de cadastro de conteúdo é necessário definir o repositório virtual de artigos científicos que será utilizado, apresentado na próxima seção.

4.2.1.2. Escolha do Repositório

O repositório escolhido para ser utilizado no trabalho é o SciELO, pois além de ser uma das bibliotecas digitais mais utilizadas, também possui um vasto número de artigos científicos em seu repositório virtual, com mais de 377 mil artigos disponíveis para consulta e *download*. O SciELO possui publicações das seguintes áreas:

- Ciências agrárias
- Ciências Sociais Aplicadas
- Ciências Biológicas
- Engenharias
- Ciências Exatas e da Terra
- Ciências da Saúde

- Ciências Humanas
- Linguística, Letras e Artes

A figura a seguir (Figura 21) apresenta a página inicial do repositório.

Figura 21 - Página inicial do SciELO

Na página inicial do SciELO é possível pesquisar por artigos, periódicos, informações sobre o site (contendo os critérios de admissão de periódicos), entre outros. A Figura 22 contém a página de resultados de uma consulta por artigos.

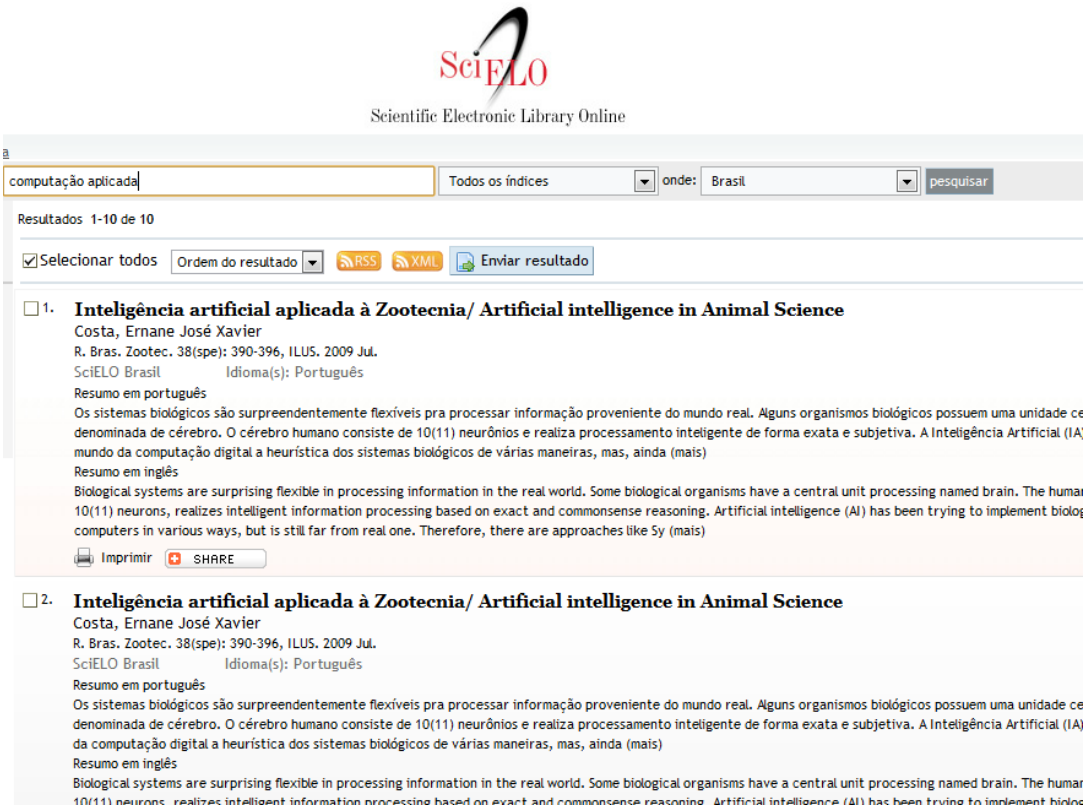


Figura 22 - Resultados de uma consulta por artigos no SciELO

Como observado na Figura 22, ao realizar uma consulta no SciELO são retornados os artigos de acordo com o termo de busca. Nesse caso, foi feita uma consulta por "computação aplicada" e o resultado retornado foi de 10 artigos relacionados ao tema; cada item do resultado contém o título, autor e resumo do artigo, juntamente com um *link* que contém todas as informações do artigo. Após ter o ambiente preparado e o repositório definido passa-se para a etapa de criação da *string* de busca; a próxima seção irá apresentará os passos necessários para a criação dessa *string*.

4.2.1.3. Criação da *String* de Busca

Para criar a *string* de busca, primeiramente foi necessário implementar um algoritmo de *stopwords* (Figura 23), para que transforme o conteúdo do material didático em uma lista de termos. O algoritmo necessita de uma *stoplist* para retirar os termos contidos nela do conteúdo, para isso foi utilizada a lista de palavras em português disponibilizada[7] pela Universidade Católica de Pelotas.

[7] <http://paginas.ucpel.tche.br/~loh/stoplists.zip> Acesso em: 26 Mar. 2012.

```
1   public List<string> StopWords(string frase)
2   {
3       List<string> palavras = removerPontuacao(frase).Split(' ').ToList();
4       List<string> remover = new List<string>();
5       Encoding enc = Encoding.GetEncoding("ISO-8859-1");
6       StreamReader red = new StreamReader(@"E:\Crawler\Arquivos\stopwords.txt", enc);
7       string Text = red.ReadToEnd();
8       using (StringReader reader = new StringReader(Text))
9       {
10          string line;
11          while ((line = reader.ReadLine()) != null)
12          {
13              foreach (string tit in palavras)
14              {
15                  if (line.Equals(tit))
16                      remover.Add(tit);
17              }
18          }
19      }
20      foreach (string item in remover)
21      {
22          palavras.Remove(item);
23      }
24      palavras.RemoveAll(c => c.Equals(""));
25      return palavras;
26  }
```

Figura 23 - Algoritmo de *StopWords*

O método “StopWords()”, apresentado na figura 23, recebe como parâmetro uma *string* contendo uma frase ou texto; na linha 3 é chamado um método que substitui pontuações como virgula e ponto final por posições vazias na *string*; logo em seguida, cada palavra da frase é transformada em um elemento de uma lista, que é retornada para uma variável “palavras” que irá receber essa lista; dessa forma tem-se uma lista onde cada elemento representa uma palavra da frase passada como parâmetro. Na linha 4 é criada uma lista de *string* que posteriormente conterá as palavras que serão removidas; as linhas 6, 7 e 8 são responsáveis por criar uma variável que referencie a *stoplist* (no formato .txt), leia o documento e o transforme em uma *string* e crie a estrutura que permita a leitura de cada linha dessa *string*. Para cada palavra da lista de *stopwords* é percorrida a lista de palavras verificando se contém alguma palavra que deverá ser removida, caso tenha, ela é adicionada a lista de palavras a remover (Linhas 10, 11, 13, 15 e 16). Após preenchida a lista de palavras que serão removidas, seus termos são retirados da lista “palavras”, retornando assim uma lista de palavras sem os termos especificados na *stoplist* (Linhas 20, 22, 24 e 25). A figura a seguir (Figura 24) apresenta o método que montará a *string* de busca.

```
public string GetConsulta()
{
    foreach (string item in StopWords(Titulo))
    {
        if (Termos.Select(a => a.Palavra.ToUpper().Equals(item.ToUpper())).Contains(true))
        {
            Termo ter = Termos.Where(a => a.Palavra.ToUpper().Equals(item.ToUpper())).SingleOrDefault();
            ter.Frequencia++;
            ter.Peso = ter.Peso + 3;
        }
        else
        {
            Termo termo = new Termo();
            termo.Palavra = item;
            termo.Frequencia = 1;
            termo.Peso = 3;
            Termos.Add(termo);
        }
    }
    foreach (string item in StopWords(Descricao))
    {
        if (Termos.Select(a => a.Palavra.ToUpper().Equals(item.ToUpper())).Contains(true))
        {
            Termo ter = Termos.Where(a => a.Palavra.ToUpper().Equals(item.ToUpper())).SingleOrDefault();
            ter.Frequencia++;
            ter.Peso++;
        }
        else
        {
            Termo termo = new Termo();
            termo.Palavra = item;
            termo.Frequencia = 1;
            termo.Peso = 1;
            Termos.Add(termo);
        }
    }
    foreach (string item in Tags)
    {
        if (Termos.Select(a => a.Palavra.ToUpper().Equals(item.ToUpper())).Contains(true))
        {
            Termo ter = Termos.Where(a => a.Palavra.ToUpper().Equals(item.ToUpper())).SingleOrDefault();
            ter.Frequencia++;
            ter.Peso = ter.Peso + 3;
        }
        else
        {
            Termo termo = new Termo();
            termo.Palavra = item;
            termo.Frequencia = 1;
            Termos.Add(termo);
            termo.Peso = 3;
            Termos.Add(termo);
        }
    }


    string retorno = "";

    List<Termo> ret = Termos.OrderByDescending(c => c.Peso).ToList();
    int contador = 0;
    int media = ret.Sum(a => a.Peso) / ret.Count;
    foreach (Termo item in ret)
    {
        if (item.Peso >= media && contador < 5)
        {
            retorno += item.Palavra + " ";
            contador++;
        }
    }
    return retorno;
}
```

Figura 24 - Método de criação da *string* de busca

O método apresentado na figura 24 é responsável por utilizar o conteúdo do material didático (título, descrição e *tags*) para montar a *string* de busca. A linha 3 cria um laço de repetição que irá executar o código para cada termo presente no retorno da chamada do método de *stopword* (Figura 23). Na linha 5 é verificado se cada termo do documento existe em uma lista de termos já verificados; caso exista, a frequência do termo é acrescida em 1 e o seu respectivo peso é acrescido de acordo com a prioridade do conteúdo (no caso do algoritmo, o título possui peso 3, descrição peso 1 e *tags* peso 3); caso não exista, o termo juntamente com seu peso e frequência é adicionado à lista de termos já verificados (Linhas 5 a 18). Esse processo é executado para o título, a descrição e as *tags* (Linhas 3 a 19, 20 a 36 e 37 a 54, respectivamente). Na linha 57 é criada uma variável "retorno" que, posteriormente, conterá a *string* de busca; Na linha 53; a variável "ret" do tipo lista de termos (linha 59) irá conter todos os termos processados anteriormente e ordenados de acordo com seu respectivo peso, em ordem decrescente. É criada uma variável "media" (Linha 61) que irá conter a soma de todos os pesos da lista, divididos pela quantidade de termos, obtendo assim uma média de pesos; logo após, é percorrida a lista de termos adicionando na variável "retorno" o termo que possui seu peso acima da média de pesos, não excedendo um limite máximo de 5 termos (Linhas 62 a 69). Após montada a *string*, a variável "retorno" é retornada (linha70) e o aplicativo está pronto pra dar inicio ao processo de *crawler*. Das figuras 25 a 29 demonstram visualmente como é feito esse processo.

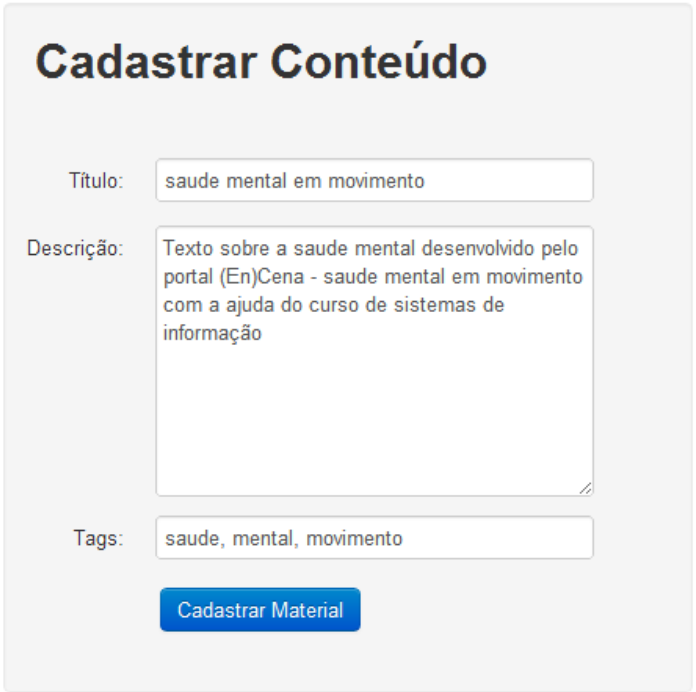


Figura 25 - Cadastro de Conteúdo

Como observado na Figura 25, o usuário insere as informações de um determinado material didático nos campos “Título”, “Descrição” e “Tags” e clica em “Cadastrar Material”. O aplicativo utiliza essas informações para dar inicio ao processo de *stopwords* (Figura 26).

Figura 26 - Processo de *StopWords*

O aplicativo retira dos campos “Título”, “Descrição” e “Tags” todas as palavras que foram definidas na *stoplist*, montando assim, uma lista de termos para cada campo (Figura 26). Após montar a lista de termos é iniciado o processo de atribuição de peso para os termos (Figura 27).

Atribuição de pesos

Título	Peso	Quantidade	Peso Final
saude	3	1	3
mental	3	1	3
movimento	3	1	3

Tags	Peso	Quantidade	Peso Final
saude	3	1	3
mental	3	1	3
movimento	3	1	3

Descrição	Peso	Quantidade	Peso Final
texto	1	1	1
saude	1	2	2
mental	1	2	2
desenvolvido	1	1	1
portal	1	1	1
(En)Cena	1	1	1
movimento	1	1	1
ajuda	1	1	1
curso	1	1	1
sistemas	1	1	1
informação	1	1	1

Figura 27 - Atribuição de pesos

Como observado na Figura 27, foi definido o peso 3 para os campos “Título” e “Tags” e peso 1 para o campo “Descrição”. O título e as *tags* possuem uma maior relevância para a recomendação pois como são campos de conteúdo sucinto, especificam de forma mais direta as características do material didático; já a descrição do arquivo possui um conteúdo de maior volume, armazenando algumas palavras que não possuem tanta relevância para a recomendação, atribuindo dessa forma um peso menor para a descrição. O aplicativo multiplica o peso do respectivo campo pela quantidade de vezes que o termo possui ocorrência, gerando assim seu peso final. Dessa forma é montada uma lista geral de termos do material didático (Figura 28).

Lista Final

Termo	Peso Final
saude	8
mental	8
movimento	7
texto	1
desenvolvido	1
portal	1
(En)Cena	1
ajuda	1
curso	1
sistemas	1
informação	1

Figura 28 - Lista final de pesos

Como observado na Figura 28, todos os termos e respectivos pesos são agrupados em uma lista ordenada pelo maior peso. Com a lista definida, é iniciada a etapa responsável por montar a *string* de busca (Figura 29).

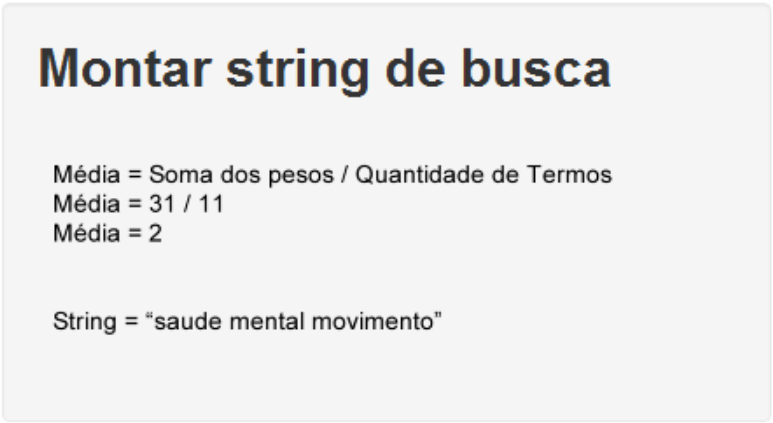


Figura 29 - Montar *string* de busca

Como observado na Figura 29, o cálculo da média resulta na divisão entre a soma de todos os pesos da lista final pela quantidade de termos da lista. O resultado é arredondado para baixo e os termos que possuam peso superior à média são utilizados na *string* de busca. Para que a *string* não fique muito extensa foi definida uma quantidade máxima de termos (cinco termos) que podem ser utilizados, no entanto essa quantidade pode vir a ser alterada caso esteja em grande ou pouca quantidade. Após montada a *string*, o aplicativo inicia o processo de *crawler*. A próxima seção irá detalhar sobre esse processo.

4.2.1.4. Crawler no repositório

A figura a seguir (Figura 30) demonstra o início do processo de *crawler*.

```
1  protected void ButtonCadastrar_Click(object sender, EventArgs e)
2  {
3      Material material = new Material(TextBoxTitulo.Text, TextBoxDescricao.Text, TextBoxTags.Text);
4      string consulta = material.GetConsulta();
5      BuscarArtigos(consulta);
6  }
```

Figura 30 - Inicio do processo de *crawler*

Para realizar o processo de *crawler*, é preciso primeiramente montar o objeto "material" (Linha 3) com os dados informados através do simulador de conteúdo. Logo após é criada uma variável do tipo *string*, que recebe o retorno da chamada do método "GetConsulta()" contendo a *string* de busca para realização do *crawler*; com isso é possível acionar o método "BuscarArtigos()" passando a *string* de busca. As figuras a seguir (Figuras 31 e 32) detalharão o funcionamento do método de *crawler*.

```
1  private static string GetHtml(string url)
2  {
3      HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
4      request.UserAgent = "Crawler";
5      WebResponse response = request.GetResponse();
6      Stream stream = response.GetResponseStream();
7      StreamReader reader = new StreamReader(stream);
8      string htmlText = reader.ReadToEnd();
9
10     return htmlText;
11 }
```

Figura 31 - Método de para obter o html de uma página web

Como observado na figura anterior (Figura 31), o método "GetHtml(url)" é responsável por obter o conteúdo html de uma página web qualquer; para tanto são realizadas requisições web (linhas 3 a 5) e transformado seu conteúdo em uma *string* (linhas e a 8),

retornando (linha 10) para o local aonde foi chamado o método. A figura a seguir (Figura 32) irá demonstrar o funcionamento do método de busca de artigos.

```
1  protected List<Artigo> BuscarArtigos(string consulta)
2  {
3      string a = GetHtml(@"http://search.scielo.org/?q="+consulta+@"&where=SCL&sort=score+desc&from=0");
4      a = a.Replace("\t", "");
5      List<string> urls = new List<string>();
6      bool verificaTitle = false;
7      bool linhaUrl = false;
8      using (StringReader reader = new StringReader(a))
9      {
10         string line;
11         while ((line = reader.ReadLine()) != null)
12         {
13             if (verificaTitle)
14             {
15                 if (linhaUrl)
16                 {
17                     string conteudo = line;
18                     conteudo = conteudo.Replace("<a href=\"", "");
19                     conteudo = conteudo.Replace("\">", "");
20                     conteudo = conteudo.Replace(" ", "");
21                     conteudo = conteudo.Replace("\t", "");
22                     urls.Add(conteudo);
23                     verificaTitle = false;
24                     linhaUrl = false;
25                 }
26                 else
27                     linhaUrl = true;
28             }
29             if (line.Contains("<!-- title -->"))
30                 verificaTitle = true;
31         }
32     }
33     return GetArtigos(urls);
34 }
```

Figura 32 - Método de recuperação de artigos

No método de recuperação de artigos (Figura 32) é possível retornar uma lista de artigos a partir de uma *string* de busca; inicialmente é criada uma variável do tipo *string* para receber o *html* com os artigos a partir da chamada do método "GetHtml()" (Linha 3). Como parâmetro do método "GetHtml()" é informada a *url* de consulta de artigos do SciELO: <http://search.scielo.org/?q="+consulta+@"&where=SCL&sort=score+desc&from=0>, sendo que pela *url* (parâmetro "from") é possível definir em qual página de artigos – em um total de 10 – que se deseja recuperar o conteúdo *html*; dessa forma para se obter o resultado de uma consulta, em que é requerido os artigos sobre câncer da posição 13 a 22, é só adicionar no parâmetro "from" à quantia "13", ficando da seguinte forma: <http://search.scielo.org/?q=cancerwhere=SCL&sort=score+desc&from=13>. Logo em seguida, são criadas variáveis (Linhas 6 e 7) que auxiliam na busca pelas *url's* dos artigos recuperados; a *string* com o *html* é percorrida (Linhas 8 a 16) até que seja localizada a linha com o referente artigo (Linha 17); a linha com a *url* é tratada de modo que apenas o endereço do artigo seja armazenado, eliminando outros caracteres desnecessários, e adicionando a uma

lista de *url's* (Linha 17 a 22). Ao final, essa lista de *url's* é utilizada como parâmetro do método "GetArtigos()", que será detalhado na figura a seguir (Figura 33).

```
1   private static List<Artigo> GetArtigos(List<string> urls)
2   {
3       List<Artigo> artigos = new List<Artigo>();
4       foreach (string url in urls)
5       {
6           try
7           {
8               string id = url.Replace("http://www.scielo.br/scielo.php?script=sci_arttext&pid=", "");
9               Artigo artigo = new Artigo();
10              artigo.url = url;
11              artigo.download = @"http://www.scielo.br/scielo.php?script=sci_pdf&pid=" + id;
12              foreach (var item in XDocument.Load(@"http://www.scielo.br/scieloOrg/php/articleXML.php?pid=" + id).Root.
13                  Elements("front").Descendants())
14              {
15                  if (item.Name.ToString().Equals("article-title") && item.LastAttribute.Value.Equals("pt"))
16                      artigo.Titulo = item.Value;
17                  if (item.Name.ToString().Equals("abstract") && item.LastAttribute.Value.Equals("pt"))
18                      artigo.resumo = item.Value;
19                  if (item.Name.ToString().Equals("kwd") && item.LastAttribute.Value.Equals("pt"))
20                      artigo.tags.Add(item.Value);
21              }
22              artigos.Add(artigo);
23          }
24          catch (Exception e)
25          {
26          }
27      }
28      return artigos;
29  }
```

Figura 33 - Método de obtenção dos conteúdos de artigos

Como visto na figura anterior (Figura 33), o método "GetArtigos()" recebe como parâmetro uma lista de *url's* e retorna uma lista de artigos com seus conteúdos completos (título, resumo, *tags* e link para download). Inicialmente é criada uma variável (Linha 3) do tipo Lista de Artigos para armazenar cada artigo da lista de *url's*; em seguida são percorridas as *url's* (Linha 4) e obtém-se seus respectivos id's (Linha 8); para cada artigo disponível no SciELO existe um arquivo *xml* com o os dados referentes ao artigo, através disso é possível percorrer esse *xml* e retirar suas informações (título, resumo e *tags*) para armazenar em um objeto do tipo "Artigo" (Linhas 9 a 21); logo após esse objeto é adicionado a lista de artigos (Linha 22) e esta é retornada pelo método (Linha 28). A figura 34 demonstra visualmente a página do SciELO após uma consulta com a *string* de busca montada.

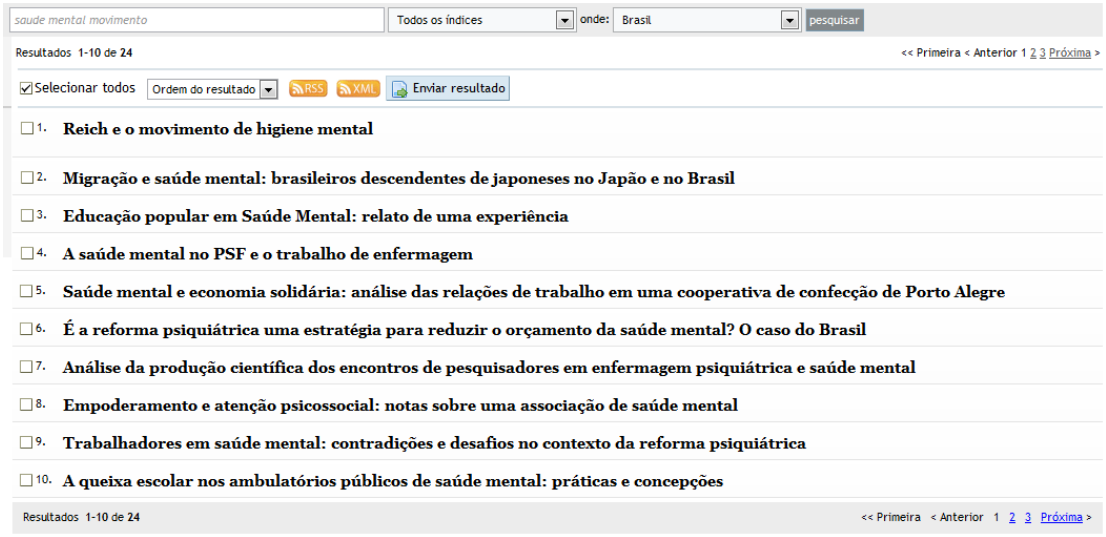


Figura 34 - Consulta por "saude mental movimento" no repositório SciELO

Como observado na Figura 34, com a *string* de busca montada na etapa anterior foi possível retornar dez artigos de um total de vinte e quatro. Dessa forma, todo o processo de *crawler* no repositório SciELO é concluído, pois já se tem a lista de artigos relacionados ao material didático. A próxima seção irá dar início a recomendação desses artigos ao material didático de acordo com sua relevância.

4.2.1.5. Recomendação

Para realizar a filtragem baseada em conteúdo é preciso primeiramente calcular o TF, o IDF e o TF-IDF de cada termo dos artigos; Para calcular o TF é necessário que seja realizado o cálculo da frequência do termo (Figura 35), utilizando o método “CalcularFrequencia()” para cada artigo recuperado.

```
1   public void CalcularFrequencia()
2   {
3       foreach (string item in StopWords(Titulo))
4       {
5           if (Termos.Select(a => a.Palavra.ToUpper().Equals(item.ToUpper())).Contains(true))
6           {
7               Termo ter = Termos.Where(a => a.Palavra.ToUpper().Equals(item.ToUpper())).SingleOrDefault();
8               ter.Frequencia++;
9           }
10          else
11          {
12              Termo termo = new Termo();
13              termo.Palavra = item;
14              termo.Frequencia = 1;
15              Termos.Add(termo);
16          }
17      }
18      if (resumo != null)
19      {
20          foreach (string item in StopWords(resumo))
21          {
22              if (Termos.Select(a => a.Palavra.ToUpper().Equals(item.ToUpper())).Contains(true))
23              {
24                  Termo ter = Termos.Where(a => a.Palavra.ToUpper().Equals(item.ToUpper())).SingleOrDefault();
25                  ter.Frequencia++;
26              }
27              else
28              {
29                  Termo termo = new Termo();
30                  termo.Palavra = item;
31                  termo.Frequencia = 1;
32                  Termos.Add(termo);
33              }
34          }
35      }
36      if (tags.Count > 0)
37      {
38          foreach (string item in tags)
39          {
40              if (Termos.Select(a => a.Palavra.ToUpper().Equals(item.ToUpper())).Contains(true))
41              {
42                  Termo ter = Termos.Where(a => a.Palavra.ToUpper().Equals(item.ToUpper())).SingleOrDefault();
43                  ter.Frequencia++;
44              }
45              else
46              {
47                  Termo termo = new Termo();
48                  termo.Palavra = item;
49                  termo.Frequencia = 1;
50                  Termos.Add(termo);
51                  Termos.Add(termo);
52              }
53          }
54      }
55  }
```

Figura 35 - Calcular frequência dos termos dos Artigos

O cálculo da frequência (Figura 35) é semelhante ao realizado no material didático. Para cada termo da lista retornada pelo método “StopWords()” (Linha 3), é feita uma busca na lista de termos do artigo (Linha 5), em que caso o termo seja encontrado, sua frequência é acrescida em 1 (Linhas 7 e 8); caso não seja encontrado, é criado um termo com uma única frequência e adicionado a lista de termos (Linhas 12 a 15). O mesmo cálculo é realizado para “resumo” e “tags”, no entanto como existem ocorrências em que essas duas informações não possuem conteúdo, é necessário verificar se existem os termos antes de efetuar o cálculo (Linhas 18 e 36). Após calcular as frequências, o algoritmo está pronto para calcular os valores de TF, IDF e TF-IDF (Figura 36).

```
1   List<Artigo> artigos = BuscarArtigos(consulta);
2
3   foreach (Artigo artigo in artigos)
4   {
5       artigo.CalcularFrequencia();
6   }
7
8   foreach (Artigo artigo in artigos)
9   {
10      foreach (Termo termo in artigo.Termos)
11      {
12          double frequenciaMaxima = artigo.Termos.OrderByDescending(c => c.Frequencia).
13              FirstOrDefault().Frequencia;
14          termo.TF = (double)termo.Frequencia / frequenciaMaxima;
15
16          double quantidadeArtigos = (double)artigos.Count;
17          double quantArtigosTermo = (double)artigos.Where(ar => ar.Termos.
18              Select(cc => cc.Palavra.Equals(termo.Palavra)).Contains(true)).Count();
19          termo.IDF = Math.Log(quantidadeArtigos / quantArtigosTermo);
20
21          termo.CalcularTFxIDF();
22      }
23  }
```

Figura 36 - Calcular TF, IDF e TF-IDF

Para calcular os valores de TF, IDF e TF-IDF são percorridos os artigos científicos da lista “artigos” (Linha 8) e para cada artigo é feito outro laço de repetição para a lista de termos do respectivo artigo (Linha 10). Primeiramente é calculada a frequência máxima de termos, ou seja, a frequência com valor mais alto (Linhas 12 e 13); logo após é calculado o TF, dividindo a frequência do termo pela frequência máxima de termos (Linha 14). Para calcular o IDF, é necessário antes calcular o número de documentos recuperados e o número de documentos que contém o termo vigente (Linhas 16 e 17); com isso é possível calcular o IDF dividindo número de documentos recuperados pelo número de documentos que contém o termo, e logo após calculando o log desse resultado (Linha 19).

Uma vez calculado o TF e o IDF, é chamado o método que calcula TF-IDF (Figura 37).

```
1   public void CalcularTFxIDF()
2   {
3       TFxIDF = TF * IDF;
4   }
```

Figura 37 - Método que calcula TF-IDF

O método apresentado na Figura 37 consiste em multiplicar o TF pelo IDF de cada termo, salvando esse resultado em uma variável, criada dentro das classes “artigo” e “material”, chamada “TFxIDF”, podendo ter seu valor acessado por quaisquer objetos dessas

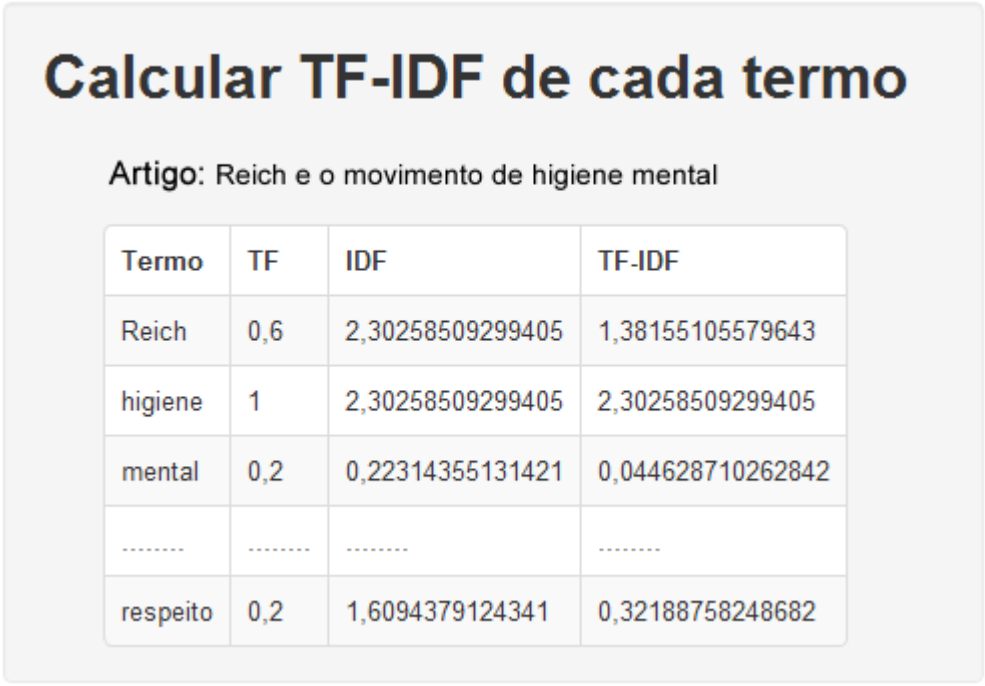
classes. Uma vez calculados os pesos, é possível calcular o cosseno do ângulo formado pelos vetores e gerar a recomendação. Esse cálculo será detalhado na Figura 38.

```
1  foreach (Artigo artigo in artigos)
2  {
3      int tamTermos = 0;
4      if (material.Termos.Count <= artigo.Termos.Count)
5          tamTermos = material.Termos.Count;
6      else
7          tamTermos = artigo.Termos.Count;
8
9      double[] vetMaterial = new double[tamTermos];
10     double[] vetArtigo = new double[tamTermos];
11
12     for (int i = 0; i < tamTermos; i++)
13     {
14         vetMaterial[i] = material.Termos[i].TFxIDF;
15         vetArtigo[i] = artigo.Termos[i].TFxIDF;
16     }
17
18     double numerador = 0;
19     double denominadorMat = 0;
20     double denominadorArt = 0;
21
22     for (int i = 0; i < tamTermos; i++)
23     {
24         numerador += vetMaterial[i] * vetArtigo[i];
25         denominadorMat += vetMaterial[i] * vetMaterial[i];
26         denominadorArt += vetArtigo[i] * vetArtigo[i];
27     }
28     double denominador = Math.Sqrt(denominadorArt) * Math.Sqrt(denominadorMat);
29     artigo.Similaridade = numerador / denominador;
30 }
31 artigos = artigos.OrderByDescending(c => c.Similaridade).ToList();
```

Figura 38 - Cálculo do cosseno

Como observado na Figura 38, primeiramente é percorrida a lista de artigos (Linha 1) e, para cada artigo, é verificado qual dos dois (artigo e material didático) possui a menor lista de termos para ser usada como parâmetro no tamanho dos vetores (Linha3 a Linha 10); após a criação dos vetores, eles são preenchidos com os respectivos pesos de cada termo (Linha 12 a Linha 16). A variável “numerador” (Linha 18) representa a soma do produto dos vetores de pesos dos termos (Linha 24); as variáveis “denominadorMat” e “denominadorArt” (Linhas 19 e 20) representam a soma do quadrado de cada peso dos termos (Linhas 25 e 26) do material didático e do artigo, respectivamente; a variável “denominador” irá conter a multiplicação entre as raízes quadradas das variáveis “denominadorMat” e “denominadorArt”; por fim, divide-se o “numerador” pelo “denominador” e o associa ao campo “Similaridade” do artigo

(Linha 29). A lista de artigos é ordenada de forma decrescente, levando em consideração o campo “Similaridade” (Linha 31). As Figuras 39 e 40 demonstrarão visualmente essa etapa.



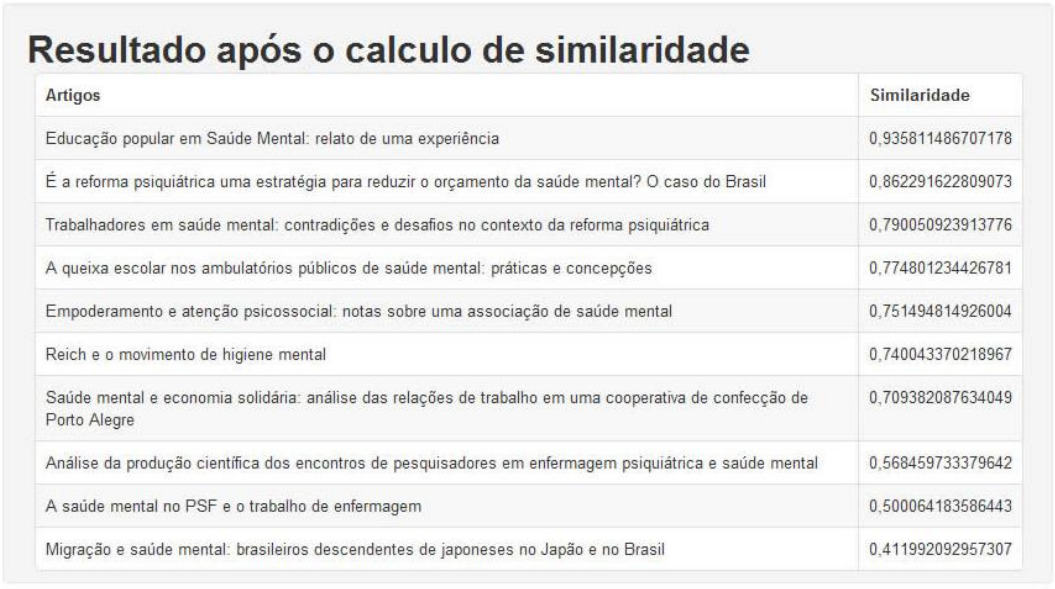
Calcular TF-IDF de cada termo

Artigo: Reich e o movimento de higiene mental

Termo	TF	IDF	TF-IDF
Reich	0,6	2,30258509299405	1,38155105579643
higiene	1	2,30258509299405	2,30258509299405
mental	0,2	0,22314355131421	0,044628710262842
--------	--------	--------	--------
respeito	0,2	1,6094379124341	0,32188758248682

Figura 39 - Calcular o TD-IDF dos termos

Como observado na Figura 39, da lista de artigos obtida pelo repositório, o aplicativo monta a lista de termos para cada artigo obtido; para cada termo são calculados o TF, IDF e TF-IDF, de acordo com as equações vistas anteriormente, fazendo com que seja possível aplicar o cálculo do cosseno e obter a similaridade dos artigos (Figura 40).



Resultado após o calculo de similaridade

Artigos	Similaridade
Educação popular em Saúde Mental: relato de uma experiência	0,935811486707178
É a reforma psiquiátrica uma estratégia para reduzir o orçamento da saúde mental? O caso do Brasil	0,862291622809073
Trabalhadores em saúde mental: contradições e desafios no contexto da reforma psiquiátrica	0,790050923913776
A queixa escolar nos ambulatórios públicos de saúde mental: práticas e concepções	0,774801234426781
Empoderamento e atenção psicossocial: notas sobre uma associação de saúde mental	0,751494814926004
Reich e o movimento de higiene mental	0,740043370218967
Saúde mental e economia solidária: análise das relações de trabalho em uma cooperativa de confecção de Porto Alegre	0,709382087634049
Análise da produção científica dos encontros de pesquisadores em enfermagem psiquiátrica e saúde mental	0,568459733379642
A saúde mental no PSF e o trabalho de enfermagem	0,500064183586443
Migração e saúde mental: brasileiros descendentes de japoneses no Japão e no Brasil	0,411992092957307

Figura 40 - Calculo de similaridade

Como observado na Figura 40, após aplicar o calculo do cosseno nos artigos recuperados, é obtida uma lista de artigos ordenados pela similaridade. Com isso o aplicativo de recomendação persiste na base de dados do Konnen os artigos e suas respectivas similaridades, os relacionando ao material didático cadastrado e exibida posteriormente a recomendação. Dessa forma, o aplicativo foi desenvolvido com o intuito de auxiliar – no que tange a recomendação de conteúdo – a rede social acadêmica Konnen, visto que será minimizado o esforço pela busca de conteúdo de estudo, aumentando a interação do Konnen com outros serviços externos a instituição.

5 CONSIDERAÇÕES FINAIS

O trabalho desenvolvido tem como finalidade oferecer um mecanismo que sirva como um recurso a mais para o auxílio na busca por conteúdo de estudo. Assim, tanto o professor quanto o acadêmico poderão de forma mais ágil encontrar fontes científicas com um nível de confiabilidade maior do que uma busca avulsa em sites diversos na Web.

Para seu desenvolvimento foi necessária a compreensão dos conceitos de Recuperação da Informação e Recomendação de Sistemas, de modo que a partir do estudo realizado foi possível definir o fluxo de ações que o aplicativo utilizaria para gerar a recomendação. Dos modelos de recuperação da informação existentes, foi escolhido o modelo espaço vetorial, por possuir uma melhor definição do quão similar um documento é do outro, visto que o modelo booleano se limita a operadores lógicos. Das abordagens de recomendação estudadas foi escolhida a Filtragem Baseada em Conteúdo, visto que não é necessário que o usuário interaja com o material didático para gerar a recomendação, utilizando-se apenas das informações do material na hora do seu cadastro.

Um dos pontos observados é a viabilidade da utilização de outros repositórios para a busca de artigos, fazendo com que o nível de confiabilidade aumente mediante testes com diversos repositórios.

Um dos problemas encontrados no decorrer do trabalho foi o fato de que o repositório escolhido inicialmente para utilização – InfoCiência – saiu do ar por um longo tempo, prejudicando o desenvolvimento do aplicativo, pois se gastou uma determinada quantia de tempo a mais procurando outro repositório e criando métodos que permitiram a extração de suas informações. Esse problema reforça mais ainda a importância de se utilizar outros repositórios de artigos científicos, para que o aplicativo não dependa exclusivamente de um único repositório, ficando a mercê de sua disponibilidade.

Como trabalhos futuros, outra abordagem para agregar valor ao aplicativo é a utilização, em conjunto, da filtragem colaborativa, que se utiliza das preferências de usuário para recomendar um conteúdo. Sua utilização seria interessante, pois é possível ser feito um trabalho de recomendação mais específico para usuários de diferentes personalidades. Essa abordagem em conjunto com a utilização de outros repositórios também pode ajudar os usuários a encontrar artigos no qual possuam uma maior afinidade, visto que ao acessar vários

artigos de um determinado repositório implicaria em uma maior preferência por ele, recomendando para o usuário artigos em maior quantidade daquele repositório ao invés de artigos de outros repositórios.

Também em trabalhos futuros é interessante que, no momento do cadastro de material didático, seja possível escolher com qual repositório, de uma lista já existente, o usuário deseja que seja feita a recomendação. Isso possibilita um melhor embasamento no momento de se recomendar um artigo, pois seria possível personalizar as recomendações de forma que repositórios específicos de cada área atendessem melhor a demanda por artigos.

Outro ponto importante como trabalhos futuros é a criação de um *Web Service* que transforme o aplicativo em um mecanismo consumido pela rede. Isso ajuda na integração das linguagens de programação, visto que o Konnen é desenvolvido em *php* e o aplicativo em questão em c#. Dessa forma o aplicativo não necessitaria ser desenvolvido na linguagem da rede e trabalharia apenas de forma que, ao receber as informações de um material didático, retornaria uma lista de artigos similares, podendo ser utilizado em quaisquer outros tipos de ambiente de desenvolvimento.

6 REFERÊNCIAS BIBLIOGRÁFICAS

AIRES, Rachel V.X. **Uso de marcadores estilísticos para a busca na Web em português**. 2005. 202 p. Tese (Doutorado em Ciências de Computação e Matemática Computacional) – USP, São Carlos.

BAEZA-YATES, Ricardo; Ribeiro-Neto, Berthier, Modern Information Retrieval, **ACM Press**, New York, USA, 1999.

BARCELLOS, Carla Duarte; MUSA, Daniela leal; BRANDÃO, André Luiz; WARPECHOWSKI, Mariusa. Sistema de Recomendação Acadêmico para Apoio a Aprendizagem. **Novas Tecnologias na Educação**, Rio Grande do Sul, v. 5, n. 2, p. 01-10, Dez. 2007.

BARION, Eliana Cristina Nogueira; LAGO, Decio. Mineração de Textos. **Revista de Ciências Exatas e Tecnologia**, São Paulo, v. 3, n. 3, p. 123-140, 8 Dez. 2008.

BRANSKI, Regina Meyer. Recuperação de informações na Web. **Perspectivas em Ciência da Informação**, Belo Horizonte, v. 9, n. 1, p. 70-87, jan./jun. 2004.

CARDOSO, Olinda Nogueira Paes. **Recuperação de Informação**. Disponível em: < http://www.dcc.ufla.br/infocomp/artigos/v2.1/art07.pdf>. Acesso em: 8 jun. 2012.

COELHO, Ricardo; AZEVEDO, Rui. **Estudo Comportamental dos Web Crawlers**. 2008. 18 p. Relatório (Disciplina de Opção III - Projecto) - FACEPE, Recife.UMINHO, Largo do Paço.

CORRÊA, Renato F. **Mapeador de teses e dissertações da UFPE (MTD-UFPE)**. 2001. 17 p. Relatório (Atividades do Projeto APQ-0728-1.03/08) - FACEPE, Recife.

FERNEDA, Edberto. **Recuperação de Informação**: Análise sobre a contribuição da Ciência da Computação para a Ciência da Informação. 2003. 147 p. Tese (Doutorado em Ciências da Comunicação) – USP, São Paulo.

GONZALEZ, Marco; LIMA, Vera L. S. de. Recuperação de Informação e Expansão Automática de Consulta com Thesaurus: uma avaliação. In: CONFERENCIA LATINOAMERICANA DE INFORMÁTICA, 27, 2001, Venezuela. **CD-ROM**. ISBN 980-110527-5. 2001. 8 p.

MONTEIRO, Lêda de Oliveira; GOMES, Igor Ruiz; OLIVEIRA, Thiago. Etapas do Processo de Mineração de Textos: uma abordagem aplicada a textos em Português do Brasil. In: Congresso da Sociedade Brasileira de Computação, 26, 2006, Campo Grande. **Anais...** Campo Grande: UFMS, 2006. p. 78-81.

PEREIRA, Diego. **Uma Aplicação em Sistemas de Recomendação**: Sistema de Recomendação para Pacotes GNU/Linux. 2007 43 p. Trabalho de Conclusão - Universidade Federal do Rio Grande do Sul, Porto Alegre.

SILVA, Altigran; MOURA, Edleno. Web Crawling: **Coleta Automática na Web**. 2002. 77 p. Disponível em: <http://www.eicstes.org/EICSTES_PDF/PRESENTATIONS/Web crawling - Coleta automática na web (Silva-Moura).pdf>. Acesso em: 05 Abr. 2011.

SOUZA, Jackson; BRITO, Parcilene; SOUSA, Cristina; SILVA, Edeilson; FAGUNDES, Fabiano; OLIVEIRA, Fernando; MARIOTI, Madianita; **Aprendizagem Organizacional Através de uma Rede de Gestão de Conhecimento**, 2012, Palmas. **Projeto de Pesquisa**... COPEX, CEULP/ULBRA.

THELWALL, Mike; STUART, David. Web Crawling Ethics Revisited: Cost, Privacy and Denial of Service. **Journal of the American Society for Information Science and Technology**, New York, v.57 n.13, p.1771-1779, Nov. 2006 .

VIEIRA, Jessica Monique de Lira; CORRÊA, Renato Fernandes. Recuperação De Informação Através De Recursos Visuais. In : Encontro Nacional de Estudantes de Biblioteconomia, Documentação, Gestão, e Ciência da Informação, 33, 2010, João Pessoa, PB. **Anais...** João Pessoa: ENEBD, 2010. Disponível em: < http://dci.ccsa.ufpb.br/enebd/index.php/enebd/article/viewFile/19/22> . Acesso em: 26 Ago. 2011.

WIVES, Leandro Krug**. Tecnologias de Descoberta de Conhecimento em Texto Aplicadas à Inteligência Competitiva**. 2002.116p. Tese (Doutorado em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <

http://pt.scribd.com/doc/35627173/TECNOLOGIAS-DE-DESCOBERTA-DE-CONHECIMENTO-EM-TEXTOS-APLICADAS-A-INTELIGENCIA-COMPETITIVA >. Acesso em: 07 Mai. 2012.